A large, light blue decorative graphic consisting of a thick curved line that forms a partial circle, with a small circle at its starting point.

8-Bit

XC82x

8-Bit Single-Chip Microcontroller

User's Manual

V1.0 2010-02

Microcontrollers

Edition 2010-02

**Published by
Infineon Technologies AG
81726 Munich, Germany**

**© 2010 Infineon Technologies AG
All Rights Reserved.**

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (www.infineon.com).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

8-Bit

XC82x

8-Bit Single-Chip Microcontroller

User's Manual

V1.0 2010-02

Microcontrollers

XC82x User's Manual**Revision History: V1.0 2010-02**

Previous Versions:

Page	Subjects (major changes since last revision)
–	

We Listen to Your Comments

Is there any information in this document that you feel is wrong, unclear or missing?
Your feedback will help us to continuously improve the quality of this document.
Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com



1	Introduction	1-1 [1]
1.1	XC82x Feature List	1-2 [1]
1.2	Pin Configuration	1-3 [1]
1.3	Pin Definitions and Functions	1-5 [1]
1.4	Chip Identification Number	1-12 [1]
1.5	Text Conventions	1-13 [1]
1.6	Reserved, Undefined and Unimplemented Terminology	1-14 [1]
1.7	Acronyms	1-14 [1]
2	XC800 Core	2-1 [1]
2.1	Overview	2-1 [1]
2.2	XC800 Core Functional Blocks	2-1 [1]
2.3	SFRs of the CPU	2-3 [1]
2.3.1	Stack Pointer (SP, 81 _H)	2-3 [1]
2.3.2	Data Pointer (DPTR, 82-3 _H)	2-3 [1]
2.3.3	Accumulator (ACC, E0 _H)	2-3 [1]
2.3.4	B Register (F0 _H)	2-3 [1]
2.3.5	Program Status Word (PSW, D0 _H)	2-3 [1]
2.3.6	Extended Operation Register (EO, A2 _H)	2-5 [1]
2.3.7	Power Control Register (PCON, 87 _H)	2-6 [1]
2.3.8	Interrupt Registers	2-6 [1]
2.4	SFRs of The Core Peripherals	2-7 [1]
2.4.1	Timer Registers	2-7 [1]
2.4.2	UART Registers	2-7 [1]
2.5	Instruction Timing	2-8 [1]
3	Memory Organization	3-1 [1]
3.1	Program Memory	3-2 [1]
3.2	Data Memory	3-2 [1]
3.2.1	Internal Data Memory	3-2 [1]
3.2.2	External Data Memory	3-2 [1]
3.3	Memory Protection Strategy	3-4 [1]
3.4	Special Function Registers	3-4 [1]
3.4.1	Address Extension by Mapping	3-4 [1]
3.4.1.1	System Control Register 0	3-6 [1]
3.4.2	Address Extension by Paging	3-7 [1]
3.4.2.1	Page Register	3-9 [1]
3.4.3	Bit-Addressing	3-10 [1]
3.4.4	Bit Protection Scheme	3-11 [1]
3.4.5	XC82x Register Overview	3-13 [1]
3.4.5.1	CPU Registers	3-13 [1]
3.4.5.2	MDU Registers	3-14 [1]
3.4.5.3	System Control Registers	3-15 [1]
3.4.5.4	Port Registers	3-18 [1]

3.4.5.5	ADC Registers	3-19 [1]
3.4.5.6	LEDSCU Registers	3-23 [1]
3.4.5.7	RTC Registers	3-24 [1]
3.4.5.8	Timer 2 Registers	3-26 [1]
3.4.5.9	CCU6 Registers	3-26 [1]
3.4.5.10	SSC Registers	3-31 [1]
3.4.5.11	IIC Registers	3-31 [1]
3.4.5.12	OCDS Registers	3-32 [1]
4	Flash Memory	4-1 [1]
4.1	Flash Memory Address Mapping	4-2 [1]
4.2	Flash Bank Sectorization	4-2 [1]
4.3	Wordline Address	4-5 [1]
4.4	Operating Modes	4-8 [1]
4.5	Error Detection and Correction	4-9 [1]
4.5.1	Flash Error Address Register	4-10 [1]
4.6	In-System Programming	4-11 [1]
4.7	In-Application Programming	4-12 [1]
4.7.1	Flash Programming	4-13 [1]
4.7.1.1	Non-background Flash Program Subroutine	4-13 [1]
4.7.1.2	Background Flash Program Subroutine	4-13 [1]
4.7.2	Flash Erasing	4-13 [1]
4.7.2.1	Non-background Flash Erase Subroutine	4-14 [1]
4.7.2.2	Background Flash Erase Subroutine	4-14 [1]
4.7.3	Aborting Background Flash Erase	4-14 [1]
4.7.4	Flash Read Mode Status	4-16 [1]
5	Boot and Startup	5-1 [1]
5.1	User Identification Number	5-2 [1]
5.2	Boot ROM Operating Mode	5-4 [1]
5.2.1	User Mode (Productive)	5-4 [1]
5.2.2	User Mode (Diagnostic)	5-4 [1]
5.2.3	Boot-Loader Mode	5-4 [1]
5.2.4	OCDS Mode	5-5 [1]
6	UART Boot-Loader	6-1 [1]
6.1	Phase I: Automatic Serial Synchronization to the Host	6-2 [1]
6.1.1	General Description	6-2 [1]
6.1.2	Calculation of BG and PRE values	6-3 [1]
6.2	Phase II: Serial Communication Protocol and the Modes	6-4 [1]
6.2.1	Serial Communication Protocol	6-4 [1]
6.2.1.1	Transfer Block Structure	6-4 [1]
6.2.1.2	Transfer Block Type	6-5 [1]
6.2.1.3	Response codes to the host	6-6 [1]

6.2.2	The Selection of Working Modes	6-8 [1]
6.2.2.1	Receiving the Header Block	6-8 [1]
6.2.2.2	Mode0: Program customer code to XRAM	6-9 [1]
6.2.2.3	Mode1: Execute customer code in XRAM	6-11 [1]
6.2.2.4	Mode2: Program customer code to FLASH	6-11 [1]
6.2.2.5	Mode3: Execute customer code in FLASH	6-13 [1]
6.2.2.6	Mode4: Erase customer code in FLASH sector(s)	6-13 [1]
6.2.2.7	Mode6: Program 4 bytes of USER_ID	6-14 [1]
6.2.2.8	ModeA: Get 4 bytes Information	6-14 [1]
7	System Control Unit	7-1 [1]
7.1	Power Supply System with Embedded Voltage Regulator	7-1 [1]
7.1.1	Reduced Voltage Condition	7-3 [1]
7.1.2	EVR Register Description	7-4 [1]
7.2	Reset Control	7-6 [1]
7.2.1	Types of Reset	7-6 [1]
7.2.1.1	Power-On Reset	7-6 [1]
7.2.1.2	Watchdog Timer Reset	7-7 [1]
7.2.1.3	Soft Reset	7-7 [1]
7.2.1.4	Power-Down Wake-Up Reset	7-7 [1]
7.2.1.5	Brownout Reset	7-7 [1]
7.2.2	Module Reset Behavior	7-8 [1]
7.2.3	Reset Control Register Description	7-9 [1]
7.3	Clock System and Control	7-11 [1]
7.3.1	Oscillator Watchdog	7-12 [1]
7.3.2	Loss of Clock Detection and Recovery	7-12 [1]
7.3.3	CCU Register Description	7-14 [1]
7.4	Power Management	7-16 [1]
7.4.1	Functional Description	7-17 [1]
7.4.1.1	Idle Mode	7-17 [1]
7.4.1.2	Power Down Mode	7-17 [1]
7.4.1.3	Peripheral Clock Management	7-21 [1]
7.4.2	Power Management Register Description	7-23 [1]
7.5	SCU Register Mapping	7-25 [1]
8	Watchdog Timer	8-1 [1]
8.1	Overview	8-1 [1]
8.2	System Information	8-2 [1]
8.2.1	Reset effects	8-2 [1]
8.2.2	Clocking Configuration	8-2 [1]
8.2.3	Interrupt Events and Assignment	8-2 [1]
8.2.4	Module Suspend Control	8-3 [1]
8.3	Functional Description	8-4 [1]
8.4	Registers Description	8-7 [1]

8.4.1	Watchdog Timer Registers	8-7 [1]
9	Interrupt System	9-1 [1]
9.1	Interrupt Sources	9-1 [1]
9.1.1	Interrupt Source and Vector	9-8 [1]
9.1.2	Interrupt Source and Priority	9-9 [1]
9.2	Interrupt Structure	9-10 [1]
9.2.1	Interrupt Structure 1	9-11 [1]
9.2.2	Interrupt Structure 2	9-11 [1]
9.3	Interrupt Handling	9-12 [1]
9.4	Interrupt Response Time	9-13 [1]
9.5	Registers Description	9-16 [1]
9.5.1	Interrupt Node Enable Registers	9-17 [1]
9.5.2	External Interrupt Control Registers	9-20 [1]
9.5.3	Interrupt Flag Registers	9-23 [1]
9.5.4	Interrupt Priority Registers	9-30 [1]
9.6	Interrupt Flag Overview	9-32 [1]
10	Debug System	10-1 [1]
10.1	Overview	10-1 [1]
10.1.1	Components of the Debug System	10-1 [1]
10.2	Product Specific Information	10-2 [1]
10.2.1	Pinning	10-2 [1]
10.2.2	Clocking Configuration	10-2 [1]
10.2.3	Interrupt Events and Assignment	10-2 [1]
10.2.4	Debug Suspend Control	10-3 [1]
10.2.5	JTAG ID	10-4 [1]
10.3	Functional Overview of the Debug System	10-5 [1]
10.3.1	Recognizing Debug-events	10-5 [1]
10.3.2	Activating the Monitor Program	10-6 [1]
10.3.3	Debug Suspend Control	10-7 [1]
10.3.4	Running the Monitor	10-7 [1]
10.3.5	Returning to the User Program	10-7 [1]
10.3.6	Single Step Execution	10-7 [1]
10.4	Breakpoint Generation Module	10-8 [1]
10.4.1	Generating Hardware Breakpoints	10-8 [1]
10.4.1.1	Breakpoints on Instruction Address	10-8 [1]
10.4.1.2	Breakpoints on Internal RAM Address	10-9 [1]
10.4.1.3	Tracing changes in Break Address	10-10 [1]
10.4.2	Processing External Breaks	10-10 [1]
10.4.3	Processing Software Breakpoints	10-10 [1]
10.5	Reactions on Breakpoints without Monitor Entry	10-11 [1]
10.5.1	Triggering a NMI request	10-11 [1]
10.6	NMI Request and Control by OCDS	10-12 [1]

10.6.1	NMI request from OCDS	10-12 [1]
10.6.2	General NMI control by OCDS	10-13 [1]
10.7	NMI-mode priority over Debug-mode	10-13 [1]
10.8	Registers Description	10-13 [1]
10.8.1	Control and Status Registers	10-15 [1]
10.8.2	Hardware Breakpoint Registers	10-16 [1]
10.8.3	Monitor Work Register	10-19 [1]
11	Parallel Ports	11-1 [1]
11.1	General Port Operation Description	11-1 [1]
11.1.1	General Register Description	11-5 [1]
11.1.1.1	Register Map	11-5 [1]
11.1.1.2	Register Overview	11-6 [1]
11.2	Port 0	11-12 [1]
11.2.1	Functions	11-12 [1]
11.2.2	Registers Description	11-17 [1]
11.3	Port 1	11-22 [1]
11.3.1	Functions	11-23 [1]
11.3.2	Registers Description	11-26 [1]
11.4	Port 2	11-30 [1]
11.4.1	Functions	11-31 [1]
11.4.2	Registers Description	11-33 [1]
12	Multiplication/Division Unit	12-1 [1]
12.1	Overview	12-1 [1]
12.2	System Information	12-2 [1]
12.2.1	Clocking Configuration	12-2 [1]
12.2.2	Interrupt Events and Assignment	12-3 [1]
12.3	Functional Description	12-3 [1]
12.3.1	Division	12-4 [1]
12.3.2	Normalize	12-5 [1]
12.3.3	Shift	12-5 [1]
12.3.4	Multiplication with Single Left Shift	12-5 [1]
12.3.5	Division with Single Right Shift	12-6 [1]
12.3.6	Busy Flag	12-6 [1]
12.3.7	Error Detection	12-6 [1]
12.4	Interrupt Generation	12-7 [1]
12.5	Registers Description	12-8 [1]
12.5.1	Operand and Result Registers	12-10 [1]
12.5.2	Control Register	12-12 [1]
12.5.3	Status Register	12-14 [1]
13	Timer 0 and Timer 1	13-1 [1]
13.1	Overview	13-1 [1]

13.2	System Information	13-1 [1]
13.2.1	Pinning	13-1 [1]
13.2.2	Clocking Configuration	13-2 [1]
13.2.3	Interrupt Events and Assignment	13-2 [1]
13.3	Basic Timer Operations	13-3 [1]
13.4	Timer Modes	13-4 [1]
13.4.1	Mode 0	13-5 [1]
13.4.2	Mode 1	13-6 [1]
13.4.3	Mode 2	13-7 [1]
13.4.4	Mode 3	13-8 [1]
13.5	Registers Description	13-9 [1]
13.5.1	Timer 0 and Timer 1 Registers	13-9 [1]
14	Timer 2	14-1 [1]
14.1	Overview	14-1 [1]
14.2	System Information	14-1 [1]
14.2.1	Pinning	14-1 [1]
14.2.2	Clocking Configuration	14-3 [1]
14.2.3	Interrupt Events and Assignment	14-3 [1]
14.2.4	IP interconnection	14-4 [1]
14.2.5	Module Suspend Control	14-4 [1]
14.3	Basic Timer Operations	14-5 [1]
14.4	Auto-Reload Mode	14-5 [1]
14.4.1	Up/Down Count Disabled	14-5 [1]
14.4.2	Up/Down Count Enabled	14-6 [1]
14.5	Capture Mode	14-8 [1]
14.6	Count Clock	14-9 [1]
14.7	External Interrupt Function	14-10 [1]
14.8	Registers Description	14-11 [1]
14.8.1	Mode Register	14-12 [1]
14.8.2	Control Register	14-13 [1]
14.8.3	Timer 2 Reload/Capture Register	14-14 [1]
14.8.4	Timer 2 Count Register	14-15 [1]
15	Real-Time Clock	15-1 [1]
15.1	Overview	15-1 [1]
15.2	System Information	15-1 [1]
15.2.1	Pinning	15-1 [1]
15.2.2	Interrupt Events and Assignment	15-1 [1]
15.2.3	Module Suspend Control	15-2 [1]
15.3	Oscillators	15-2 [1]
15.4	Basic Timer Operation	15-2 [1]
15.5	Real-Time Clock Modes	15-2 [1]
15.5.1	Mode 1: Periodic Wake-up Mode with 75 KHz Oscillator Clock ..	15-3 [1]

15.5.2	Mode 3: Timer Mode with External Clock	15-4 [1]
15.6	Power Saving Mode Option	15-5 [1]
15.7	Registers Description	15-6 [1]
15.7.1	Real-Time Clock Registers	15-7 [1]
16	UART	16-1 [1]
16.1	Overview	16-1 [1]
16.2	System Information	16-1 [1]
16.2.1	Pinning	16-1 [1]
16.2.2	Clocking Configuration	16-2 [1]
16.2.3	Interrupt Events and Assignment	16-3 [1]
16.3	UART Modes	16-3 [1]
16.3.1	Mode 0, 8-Bit Shift Register, Fixed Baud Rate	16-3 [1]
16.3.2	Mode 1, 8-Bit UART, Variable Baud Rate	16-4 [1]
16.3.3	Mode 2, 9-Bit UART, Fixed Baud Rate	16-6 [1]
16.3.4	Mode 3, 9-Bit UART, Variable Baud Rate	16-6 [1]
16.4	Multiprocessor Communication	16-8 [1]
16.5	Baud Rate Generation	16-9 [1]
16.5.1	Fixed Clock	16-9 [1]
16.5.2	UART Baud-rate Generator	16-9 [1]
16.5.3	Timer 1	16-12 [1]
16.6	LIN Support in UART	16-13 [1]
16.6.1	LIN Protocol	16-13 [1]
16.6.2	LIN Header Transmission	16-15 [1]
16.6.2.1	Automatic Synchronization to the Host	16-15 [1]
16.6.2.2	Initialization of Break/Synch Field Detection Logic	16-16 [1]
16.6.2.3	Baud Rate Range Selection	16-16 [1]
16.6.2.4	LIN Baud Rate Detection	16-18 [1]
16.7	Registers Description	16-19 [1]
16.7.1	UART Registers	16-20 [1]
16.7.2	Baud-rate Generator Control and Status Registers	16-22 [1]
16.7.3	Baud-rate Generator Timer/Reload Registers	16-24 [1]
17	Inter-IC Bus	17-1 [1]
17.1	Overview	17-1 [1]
17.2	System Information	17-1 [1]
17.2.1	Pinning	17-1 [1]
17.2.1.1	Output Pin Configuration	17-2 [1]
17.2.2	Clocking Configuration	17-2 [1]
17.2.3	Interrupt Events and Assignment	17-3 [1]
17.3	Status Code	17-4 [1]
17.4	Baud Rate Generation	17-5 [1]
17.5	Clock Synchronization	17-6 [1]
17.6	Bus Arbitration	17-6 [1]

17.7	Software Reset	17-7 [1]
17.8	Operating Modes	17-7 [1]
17.8.1	Master Transmit	17-7 [1]
17.8.2	Master Receive	17-10 [1]
17.8.3	Slave Transmit	17-12 [1]
17.8.4	Slave Receive	17-13 [1]
17.9	Registers Description	17-15 [1]
17.9.1	Slave Address Registers	17-16 [1]
17.9.2	Data Register	17-17 [1]
17.9.3	Control Register	17-18 [1]
17.9.4	Status Register	17-20 [1]
17.9.5	Baud Rate Control Register	17-21 [1]
17.9.6	Software Reset Register	17-21 [1]
18	High-Speed Synchronous Serial Interface	18-1 [1]
18.1	Overview	18-1 [1]
18.2	System Information	18-2 [1]
18.2.1	Pinning	18-2 [1]
18.2.2	Clocking Configuration	18-6 [1]
18.2.3	Interrupt Events and Assignment	18-7 [1]
18.3	General Operation	18-8 [1]
18.3.1	Operating Mode Selection	18-8 [1]
18.3.2	Full-Duplex Operation	18-10 [1]
18.3.3	Half-Duplex Operation	18-13 [1]
18.3.4	Continuous Transfers	18-14 [1]
18.3.5	Baud Rate Generation	18-15 [1]
18.3.6	Error Detection Mechanisms	18-16 [1]
18.4	Interrupts	18-19 [1]
18.5	Register Description	18-20 [1]
18.5.1	Configuration Register	18-21 [1]
18.5.2	Baud Rate Timer Reload Register	18-26 [1]
18.5.3	Transmitter Buffer Register	18-26 [1]
18.5.4	Receiver Buffer Register	18-27 [1]
19	LED and Touch-Sense Controller	19-1 [1]
19.1	Overview	19-1 [1]
19.2	System Information	19-2 [1]
19.2.1	Pinning	19-2 [1]
19.2.2	Clocking Configuration	19-3 [1]
19.2.3	Interrupt Events and Assignment	19-4 [1]
19.2.4	IP Interconnection	19-4 [1]
19.2.5	Debug Suspend Control	19-5 [1]
19.3	Time-Multiplexed LED & Touch-Sense Functions On Pin	19-5 [1]
19.4	LED Driving	19-8 [1]

19.4.1	LED Pin Assignment and Current Capability	19-10 [1]
19.5	Touchpad Sensing	19-11 [1]
19.5.1	Finger Sensing	19-13 [1]
19.6	Time-Multiplexed LED and Touch-Sense Function on Pin	19-14 [1]
19.7	Function Enabling and Control Hints	19-15 [1]
19.8	LEDTSCU Timing Calculations	19-16 [1]
19.9	LEDTSCU Pin Control	19-18 [1]
19.10	Interrupt	19-19 [1]
19.11	Registers Description	19-20 [1]
19.11.1	Global Control and Status	19-21 [1]
19.11.2	Function Control Registers	19-24 [1]
20	Capture/Compare Unit 6 (CCU6)	20-1 [1]
20.1	Introduction	20-1 [1]
20.1.1	Feature Set Overview	20-2 [1]
20.1.2	Block Diagram	20-3 [1]
20.1.3	Register Overview	20-4 [1]
20.2	System Information	20-8 [1]
20.2.1	Pinning	20-8 [1]
20.2.2	Clocking Configuration	20-12 [1]
20.2.3	Interrupt Events and Assignment	20-13 [1]
20.2.4	IP Interconnection	20-15 [1]
20.2.5	Module Suspend Control	20-17 [1]
20.3	Operating Timer T12	20-17 [1]
20.3.1	T12 Overview	20-19 [1]
20.3.2	T12 Counting Scheme	20-21 [1]
20.3.2.1	Clock Selection	20-21 [1]
20.3.2.2	Edge-Aligned / Center-Aligned Mode	20-21 [1]
20.3.2.3	Single-Shot Mode	20-24 [1]
20.3.3	T12 Compare Mode	20-25 [1]
20.3.3.1	Compare Channels	20-25 [1]
20.3.3.2	Channel State Bits	20-26 [1]
20.3.3.3	Hysteresis-Like Control Mode	20-31 [1]
20.3.4	Compare Mode Output Path	20-32 [1]
20.3.4.1	Dead-Time Generation	20-32 [1]
20.3.4.2	State Selection	20-34 [1]
20.3.4.3	Output Modulation and Level Selection	20-35 [1]
20.3.5	T12 Capture Modes	20-37 [1]
20.3.6	T12 Shadow Register Transfer	20-41 [1]
20.3.7	Timer T12 Operating Mode Selection	20-42 [1]
20.3.8	T12 related Registers	20-43 [1]
20.3.8.1	T12 Counter Register	20-43 [1]
20.3.8.2	Period Register	20-44 [1]

20.3.8.3	Capture/Compare Registers	20-46 [1]
20.3.8.4	Capture/Compare Shadow Registers	20-47 [1]
20.3.8.5	Dead-time Control Register	20-49 [1]
20.3.9	Capture/Compare Control Registers	20-51 [1]
20.3.9.1	Channel State Bits	20-51 [1]
20.3.9.2	T12 Mode Control Register	20-55 [1]
20.3.9.3	Timer Control Registers	20-57 [1]
20.4	Operating Timer T13	20-67 [1]
20.4.1	T13 Overview	20-67 [1]
20.4.2	T13 Counting Scheme	20-70 [1]
20.4.2.1	T13 Counting	20-70 [1]
20.4.2.2	Single-Shot Mode	20-71 [1]
20.4.2.3	Synchronization to T12	20-72 [1]
20.4.3	T13 Compare Mode	20-74 [1]
20.4.4	Compare Mode Output Path	20-76 [1]
20.4.5	T13 Shadow Register Transfer	20-77 [1]
20.4.6	T13 related Registers	20-79 [1]
20.4.6.1	T13 Counter Register	20-79 [1]
20.4.6.2	Period Register	20-81 [1]
20.4.6.3	Compare Register	20-83 [1]
20.4.6.4	Compare Shadow Register	20-83 [1]
20.5	Trap Handling	20-85 [1]
20.6	Multi-Channel Mode	20-87 [1]
20.7	Hall Sensor Mode	20-90 [1]
20.7.1	Hall Pattern Evaluation	20-91 [1]
20.7.2	Hall Pattern Compare Logic	20-93 [1]
20.7.3	Hall Mode Flags	20-94 [1]
20.7.4	Hall Mode for Brushless DC-Motor Control	20-96 [1]
20.8	Modulation Control Registers	20-98 [1]
20.8.1	Modulation Control	20-98 [1]
20.8.2	Trap Control Register	20-100 [1]
20.8.3	Passive State Level Register	20-103 [1]
20.8.4	Multi-Channel Mode Registers	20-104 [1]
20.9	Interrupt Handling	20-110 [1]
20.9.1	Interrupt Structure	20-110 [1]
20.9.2	Interrupt Registers	20-112 [1]
20.9.2.1	Interrupt Status Register	20-112 [1]
20.9.2.2	Interrupt Status Set Register	20-116 [1]
20.9.2.3	Status Reset Register	20-118 [1]
20.9.2.4	Interrupt Enable Register	20-120 [1]
20.9.2.5	Interrupt Node Pointer Register	20-124 [1]
20.10	General Module Operation	20-126 [1]
20.10.1	Input Selection	20-126 [1]

20.10.2	General Registers	20-127 [1]
20.10.2.1	Port Input Select Registers	20-127 [1]
20.11	Register Mapping	20-131 [1]
21	Analog to Digital Converter	21-1 [1]
21.1	System Information	21-3 [1]
21.1.1	Pinning	21-3 [1]
21.1.2	Clocking Configuration	21-3 [1]
21.1.3	Interrupt Events and Assignment	21-4 [1]
21.1.4	IP Interconnection	21-4 [1]
21.2	Introduction and Basic Structure	21-7 [1]
21.3	Electrical Models	21-10 [1]
21.4	Transfer Characteristics and Error Definitions	21-12 [1]
21.5	Configuration of General Functions	21-13 [1]
21.5.1	General Clocking Scheme and Control	21-13 [1]
21.6	Conversion Request Generation	21-17 [1]
21.6.1	Channel Scan Request Source Handling	21-19 [1]
21.6.2	Queued Request Source Handling	21-25 [1]
21.6.3	Hardware Trigger Selection	21-37 [1]
21.7	Request Source Arbitration	21-38 [1]
21.7.1	Arbiter Timing	21-40 [1]
21.7.2	Request Source Priority and Conversion Start Mode	21-41 [1]
21.8	Analog Input Channel Configuration	21-45 [1]
21.8.1	Reference Selection	21-45 [1]
21.8.2	Channel Parameters	21-48 [1]
21.8.3	Limit Checking	21-52 [1]
21.8.4	Alias Feature	21-54 [1]
21.8.5	Out of Range Comparator	21-57 [1]
21.8.6	Conversion Timing	21-61 [1]
21.8.7	Channel Events and Interrupts	21-63 [1]
21.9	Conversion Result Handling	21-64 [1]
21.9.1	Storage of Conversion Results	21-64 [1]
21.10	Wait-for-Read Mode	21-79 [1]
21.10.1	Result Events and Interrupts	21-80 [1]
21.10.2	Data Reduction and Filtering	21-80 [1]
21.11	Interrupt Request Handling	21-83 [1]
21.12	Register Mapping	21-89 [1]
22	Boot ROM User Routines	22-1 [1]
22.1	Flash Bank Read Mode Status Subroutine	22-2 [1]
22.2	Get 4 Bytes Information	22-3 [1]
22.3	Feature Setting Subroutine	22-4 [1]
22.4	UART Auto Baud Subroutine	22-5 [1]
22.5	UART BSL Routine	22-6 [1]

22.6	Flash Program Subroutine	22-7 [1]
22.7	Flash Erase Subroutine	22-9 [1]
22.8	Abort Flash Erase Subroutine	22-11 [1]
23	ROM Library	23-1 [1]
23.1	Fixed Point ROM Library	23-3 [1]
23.1.1	P Controller Routine	23-3 [1]
23.1.2	PI Controller Routine	23-4 [1]
23.1.3	PT1_24 Controller Routine	23-6 [1]
23.1.4	PT1_32 Controller Routine	23-7 [1]
23.1.5	Clarke Transform Routine	23-8 [1]
23.2	LED and Touch-Sense Controller ROM Library	23-10 [1]
23.2.1	SET_LDLINE_CMP Function (LED and TS)	23-11 [1]
23.2.1.1	Inputs for SET_LDLINE_CMP Function (LED only)	23-13 [1]
23.2.1.2	Inputs for SET_LDLINE_CMP Function (Touch-sense only)	23-15 [1]
23.2.1.3	Inputs for SET_LDLINE_CMP Function (LED and TS)	23-17 [1]
23.2.2	FINDTOUCHEDPAD Function (TS)	23-21 [1]
23.2.2.1	Outputs of Function	23-27 [1]
23.2.2.2	Implementation Details of Function	23-30 [1]
23.2.3	Use of the functions in Interrupts	23-37 [1]
23.3	MDU ROM Library (MATH Function)	23-39 [1]
23.3.1	Integer Multiplication	23-39 [1]
23.3.2	Long Multiplication	23-40 [1]
23.3.3	Integer Division	23-41 [1]
23.3.4	Long Division	23-42 [1]
23.4	EEPROM Emulation ROM Library	23-44 [1]
23.4.1	Feature	23-44 [1]
23.4.2	System requirement	23-44 [1]
23.4.3	Concept	23-44 [1]
23.4.4	API description	23-46 [1]
23.4.4.1	Constant definition	23-46 [1]
23.4.4.2	EEPROMInfo data structure	23-46 [1]
23.4.4.3	Functions	23-47 [1]
23.4.4.4	Example of API usage	23-50 [1]

1 Introduction

The XC82x is a member of the high-performance XC800 family of 8-bit microcontrollers. It is based on the XC800 Core that is compatible with the industry standard 8051 processor. The XC82x features a great number of enhancements to enable new application technologies through its highly integrated on-chip components, such as on-chip oscillator or an integrated voltage regulator, allowing a range of voltage supply of 2.5 V to 5.5 V. In addition, the XC82x is equipped with embedded Flash memory to offer high flexibility in development and ramp-up. The XC82x memory protection strategy features read-out protection of user intellectual property (IP).

Other key features include a Capture/Compare Unit 6 (CCU6) for the generation of pulse width modulated signal with special modes for motor control; a 10-bit Analog-to-Digital Converter (ADC) with out of range comparator that has differential input channels and extended functionalities such as autoscan and result accumulation for anti-aliasing filtering or for averaging; a Multiplication/Division Unit (MDU) to support the XC800 Core in math-intensive computations in advanced motor control like Field Oriented Control; a LED and Touch-sense Controller (LEDTSCU) for driving 7 segment displays in a LED matrix and supporting touch-sense function concurrently ; a Real Time Clock (RTC) to support periodic wake-up in low power application and an On-Chip Debug Support (OCDS) unit for software development and debugging of XC800-based systems. Local Interconnect Network (LIN) applications are also supported through extended UART features and the provision of LIN low level drivers for most devices. For low power applications, various power saving modes are available for selection by the user. Control of the numerous on-chip peripheral functionalities is achieved by extending the Special Function Register (SFR) address range with an intelligent paging mechanism optimized for interrupt handling.

Figure 1-1 shows the functional units of the XC82x.

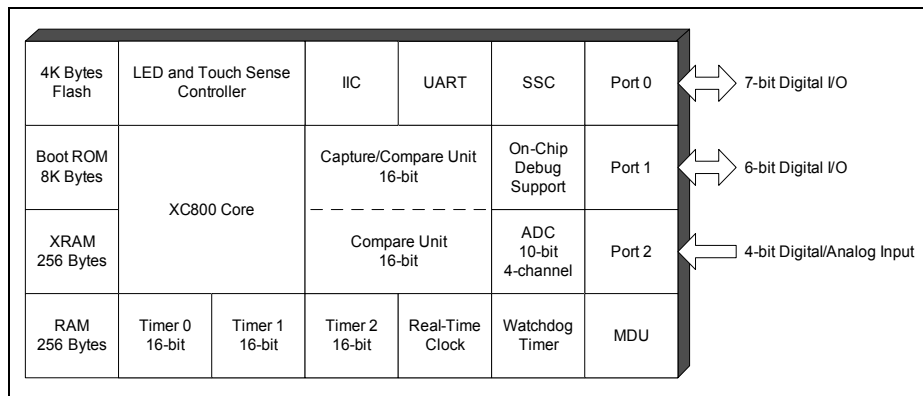


Figure 1-1 XC82x Functional Units

1.1 XC82x Feature List

The following list summarizes the main features of the XC82x:

- High performance XC800 Core
 - compatible to standard 8051 Core
 - two clocks per machine cycle architecture (for memory access without wait state)
 - two data pointers
- On-chip Memory
 - 8-KByte Boot ROM for startup firmware, Flash and User routines and ROM library
 - 256-byte RAM; plus 64-byte Monitor RAM
 - 256-byte XRAM
 - 4-KByte Flash for program code and data; (includes memory protection strategy)
- I/O port supply at 2.5 - 5.5V and core logic supply at 2.5V (generated by embedded voltage regulator)
- Power-on reset generation
- Brown-out detection for IO supply and core logic supply
- On-chip OSC for clock generation
 - Loss-of-clock detection
- Power saving modes
 - idle mode
 - power-down mode with wake-up capability via real-time clock interrupt
 - clock gating control to each peripheral
- Watchdog Timer (WDT) with programmable window feature for refresh operation and warning prior to overflow
- Three general purpose I/O ports
 - Up to 17 pins as digital I/O
 - 4 pin as digital/analog input
- Multiplication/Division Unit (MDU) for arithmetic calculation
- Up to 4 channels, 10-bit A/D Converter
 - support up to 3 differential input channel
- Up to 4 channels, Out of range comparator
- Three 16-bit timers - Timer 0, Timer 1, Timer2
- Periodic wake-up timer
- Capture/compare unit for PWM signal generation (CCU6)
- Full duplex serial interface (UART)
- Synchronous serial channel (SSC)
- Inter-IC (IIC) serial interface
- LED and Touch-sense Controller (LEDTSCU)
- On-chip Debug Support via single pin DAP interface (SPD)
 - 1-KByte monitor ROM (part of the Boot ROM)
 - 64-byte monitor RAM
- PG-TSSOP-16 and PG-DSO-20 pin packages

The block diagram of the XC82x is shown in [Figure 1-2](#).

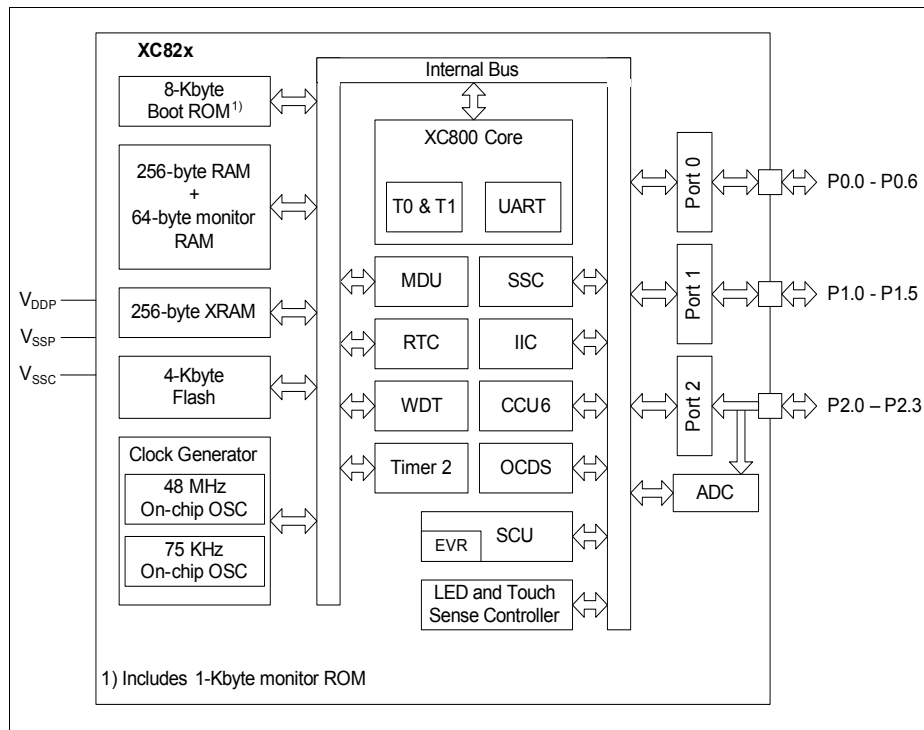


Figure 1-2 XC82x Block Diagram

1.2 Pin Configuration

Figure 1-3 shows the pin configuration of XC824 in DSO-20 package and **Figure 1-4** shows the pin configuration of XC822 in TSSOP-16 package.

Introduction

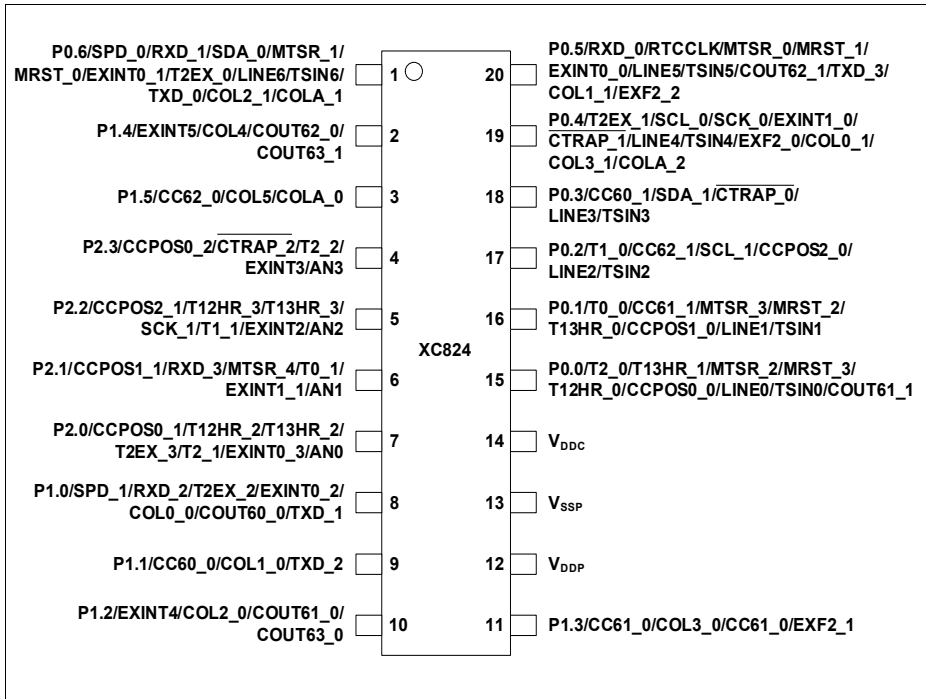


Figure 1-3 XC824 (DSO-20 package) Pin Configuration (Top View)

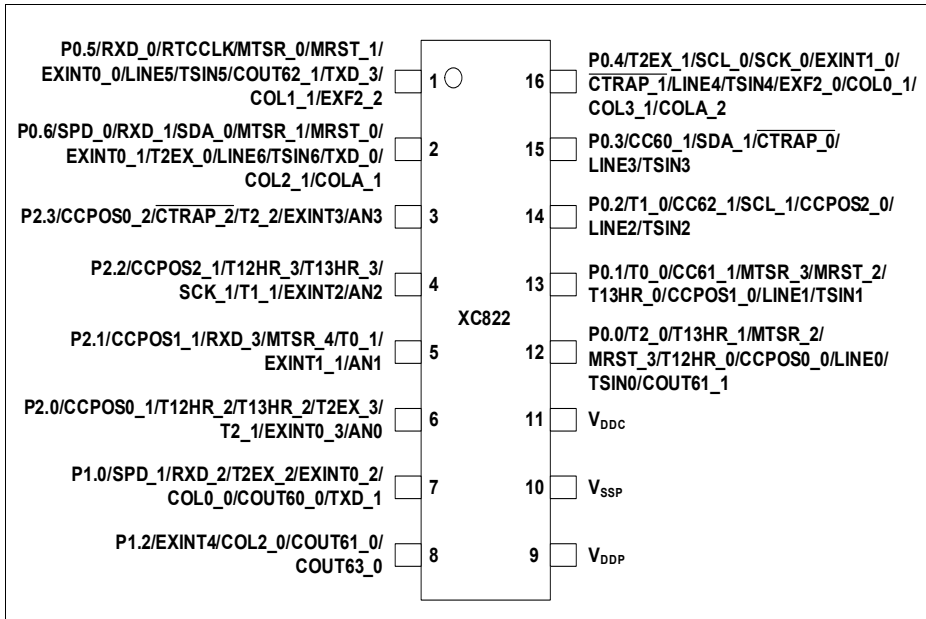


Figure 1-4 XC822 (TSSOP-16 package) Pin Configuration (Top View)

1.3 Pin Definitions and Functions

After reset, all pins are configured as input with one of the following:

- Pull-up device enabled only (PU)
- Pull-down device enabled only (PD)
- High impedance with both pull-up and pull-down devices disabled (Hi-Z)

The functions and default states of the XC82x external pins are provided in [Table 1-1](#).

Table 1-1 Pin Definitions and Functions for XC82x

Symbol	Pin Number DSO20/ TSSOP16	Type	Reset State	Function
P0		I/O		Port 0 Port 0 is a bidirectional general purpose I/O port. It can be used as alternate functions for LEDTSCU, Timer 0, 1 and 2, SSC, CCU6, IIC, SPD and UART.
P0.0	15/12		Hi-Z	<div>T2_0 Timer 2 Input</div> <div>T13HR_1 CCU6 Timer 13 Hardware Run Input</div> <div>MTSR_2 SSC Master Transmit Output/ Slave Receive Input</div> <div>MRST_3 SSC Master Receive Input</div> <div>T12HR_0 CCU6 Timer 12 Hardware Run Input</div> <div>CCPOS0_0 CCU6 Hall Input 0</div> <div>TSIN0 Touch-sense Input 0</div> <div>LINE0 LED Line 0</div> <div>COUT61_1 Output of Capture/Compare Channel 1</div>
P0.1	16/13		Hi-Z	<div>T0_0 Timer 0 Input</div> <div>CC61_1 Input/Output of Capture/Compare channel 1</div> <div>MTSR_3 SSC Slave Receive Input</div> <div>MRST_2 SSC Master Receive Input/ Slave Transmit Output</div> <div>T13HR_0 CCU6 Timer 13 Hardware Run Input</div> <div>CCPOS1_0 CCU6 Hall Input 1</div> <div>TSIN1 Touch-sense Input 1</div> <div>LINE1 LED Line 1</div>

Table 1-1 Pin Definitions and Functions for XC82x

Symbol	Pin Number DSO20/ TSSOP16	Type	Reset State	Function	
P0.2	17/14		Hi-Z	T1_0	Timer 1 Input
				CC62_1	Input/Output of Capture/Compare channel 1
				SCL_1	IIC Clock Line
				CCPOS2_0	CCU6 Hall Input 2
				TSIN2	Touch-sense Input 2
				LINE2	LED Line 2
P0.3	18/15		Hi-Z	CC60_1	Input/Output of Capture/Compare channel 1
				SDA_1	IIC Data Line
				$\overline{\text{CTRAP}}_0$	CCU6 Trap Input
				TSIN3	Touch-sense Input 3
				LINE3	LED Line 3
P0.4	19/16		PD	T2EX_1	Timer 2 External Trigger Input
				SCK_0	SSC Clock Input/Output
				SCL_0	IIC Clock Line
				$\overline{\text{CTRAP}}_1$	CCU6 Trap Input
				EXINT1_0	External Interrupt Input 1
				TSIN4	Touch-sense Input 4
				LINE4	LED Line 4
				EXF2_0	Timer 2 Overflow Flag
				COL0_1	LED Column 0
				COL3_1	LED Column 3
				COLA_2	LED Column A

Table 1-1 Pin Definitions and Functions for XC82x

Symbol	Pin Number DSO20/ TSSOP16	Type	Reset State	Function	
P0.5	20/1		Hi-Z	RXD_0	UART Receive Input
				RTCCLK	RTC External Clock Input
				MTSR_0	SSC Master Transmit Output/ Slave Receive Input
				MRST_1	SSC Master Receive Input
				EXINT0_0	External Interrupt Input 0
				TSIN5	Touch-sense Input 5
				LINE5	LED Line 5
				COUT62_1	Output of Capture/Compare Channel 2
				TXD_3	UART Transmit Output/ 2-wire UART BSL Transmit Output
				COL1_1	LED Column 1
				EXF2_2	Timer 2 Overflow Flag
P0.6	1/2		PU	SPD_0	SPD Input/Output
				RXD_1	UART Receive Input/ UART BSL Receive Input
				SDA_0	IIC Data Line
				MTSR_1	SSC Slave Receive Input
				MRST_0	SSC Master Receive Input/ Slave Transmit Output
				EXINT0_1	External Interrupt Input 0
				T2EX_0	Timer 2 External Trigger Input
				TSIN6	Touch-sense Input 6
				LINE6	LED Line 6
				TXD_0	UART Transmit Output/ 1-wire UART BSL Transmit Output
				COL2_1	LED Column 2
				COLA_1	LED Column A

Table 1-1 Pin Definitions and Functions for XC82x

Symbol	Pin Number DSO20/ TSSOP16	Type	Reset State	Function
P1		I/O		Port 1 Port 1 is a bidirectional general purpose I/O port. It can be used as alternate functions for CCU6, LEDTSCU, SPD, UART and Timer 2.
P1.0	8/7		Hi-Z	SPD_1 SPD Input/Output RXD_2 UART Receive Input T2EX_2 Timer 2 External Trigger Input EXINT0_2 External Interrupt Input 0 COL0_0 LED Column 0 COUT60_0 Output of Capture/Compare Channel 0 TXD_1 UART Transmit Output
P1.1	9/-		Hi-Z	CC60_0 Input/Output of Capture/Compare channel 0 COL1_0 LED Column 1 TXD_2 UART Transmit Output
P1.2	10/8		Hi-Z	EXINT4 External Interrupt Input 4 COL2_0 LED Column 2 COUT61_0 Output of Capture/Compare channel 1 COUT63_0 Output of Capture/Compare channel 3
P1.3	11/-		Hi-Z	CC61_0 Input/Output of Capture/Compare channel 1 COL3_0 LED Column 3 EXF2_1 Timer 2 Overflow Flag

Table 1-1 Pin Definitions and Functions for XC82x

Symbol	Pin Number DSO20/ TSSOP16	Type	Reset State	Function	
P1.4	2/-		Hi-Z	EXINT5	External Interrupt Input 5
				COL4	LED Column 4
				COUT62_0	Output of Capture/Compare channel 2
				COUT63_1	Output of Capture/Compare channel 3
P1.5	3/-		Hi-Z	CC62_0	Input/Output of Capture/Compare channel 2
				COL5	LED Column 5
				COLA_0	LED Column A
P2		I		Port 2 Port 2 is a general purpose input-only port. It can be used as inputs for A/D Converter and out of range comparator, CCU6, Timer 2, SSC and UART.	
P2.0	7/6		Hi-Z	CCPOS0_1	CCU6 Hall Input 0
				T12HR_2	CCU6 Timer 12 Hardware Run Input
				T13HR_2	CCU6 Timer 13 Hardware Run Input
				T2EX_3	Timer 2 External Trigger Input
				T2_1	Timer 2 Input
				EXINT0_3	External Interrupt Input 0
				AN0	Analog Input 0 / Out of range comparator channel 0

Table 1-1 Pin Definitions and Functions for XC82x

Symbol	Pin Number DSO20/ TSSOP16	Type	Reset State	Function
P2.1	6/5		Hi-Z	CCPOS1_1 CCU6 Hall Input 1 RXD_3 UART Receive Input MTSR_4 Slave Receive Input T0_1 Timer 0 Input EXINT1_1 External Interrupt Input 1 AN1 Analog Input 1 / Out of range comparator channel 1
P2.2	5/4		Hi-Z	CCPOS2_1 CCU6 Hall Input 2 T12HR_3 CCU6 Timer 12 Hardware Run Input T13HR_3 CCU6 Timer 13 Hardware Run Input SCK_1 SSC Clock Input/Output T1_1 Timer 1 Input EXINT2 External Interrupt Input 2 AN2 Analog Input 2 / Out of range comparator channel 2
P2.3	4/3		Hi-Z	CCPOS0_2 CCU6 Hall Input 0 CTRAP_2 CCU6 Trap Input T2_2 Timer 2 Input EXINT3 External Interrupt Input 3 AN3 Analog Input 3 / Out of range comparator channel 3
V _{DDP}	12/9	—		I/O Port Supply (2.5 V - 5.5 V)
V _{DDC}	14/11	—		Core Supply Output (2.5 V)
V _{SSP} / V _{SSC}	13/10	—		I/O Port Ground/ Core Supply Ground

1.4 Chip Identification Number

Each device variant of XC82x is assigned an unique chip identification number to allow easy identification of one device variant from the others. The differentiation is based on the product, variant type and device step information.

Two methods are provided to read a device variant's chip identification number:

- In-application subroutine, see [Chapter 22.2](#);
- Boot-loader (BSL) mode A, see [Chapter 6.2.2.8](#).

1.5 Text Conventions

This document uses the following text conventions for named components of the XC82x:

- Functional units of the XC82x are shown in upper case. For example: “The SSC can be used to communicate with shift registers.”
- Pins using negative logic are indicated by an overbar. For example: “A reset input pin $\overline{\text{RESET}}$ is provided for the hardware reset.”
- Bit fields and bits in registers are generally referenced as “Register name.Bit field” or “Register name.Bit”. Most of the register names contain a module name prefix, separated by an underscore character “_” from the actual register name. In the example of “SSC_CON”, “SSC” is the module name prefix, and “CON” is the actual register name).
- Variables that are used to represent sets of processing units or registers appear in mixed-case type. For example, the register name “CC6xR” refers to multiple “CC6xR” registers with the variable x (x = 0, 1, 2). The bounds of the variables are always specified where the register expression is first used (e.g., “x = 0 - 2”), and is repeated as needed.
- The default radix is decimal. Hexadecimal constants have a suffix with the subscript letter “H” (e.g., C0_H). Binary constants have a suffix with the subscript letter “B” (e.g., 11_B).
- When the extents of register fields, groups of signals, or groups of pins are collectively named in the body of the document, they are represented as “NAME[A:B]”, which defines a range, from B to A, for the named group. Individual bits, signals, or pins are represented as “NAME[C]”, with the range of the variable C provided in the text (e.g., CFG[2:0] and TOS[0]).
- Units are abbreviated as follows:
 - **MHz** = Megahertz
 - **μs** = Microseconds
 - **kBaud, kbit** = 1000 characters/bits per second
 - **MBaud, Mbit** = 1,000,000 characters/bits per second
 - **Kbyte** = 1024 bytes of memory
 - **Mbyte** = 1,048,576 bytes of memory

In general, the k prefix scales a unit by 1000 whereas the K prefix scales a unit by 1024. Hence, the Kbyte unit scales the expression preceding it by 1024. The kBaud unit scales the expression preceding it by 1000. The M prefix scales by 1,000,000 or 1048576, and μ scales by 0.000001. For example, 1 Kbyte is 1024 bytes, 1 Mbyte is 1024 × 1024 bytes, 1 kBaud/kbit are 1000 characters/bits per second, 1 MBaud/Mbit are 1,000,000 characters/bits per second, and 1 MHz is 1,000,000 Hz.
- Data format quantities are defined as follows:
 - **Byte** = 8-bit quantity

1.6 Reserved, Undefined and Unimplemented Terminology

In tables where register bit fields are defined, the following conventions are used to indicate undefined and unimplemented function. Further, types of bits and bit fields are defined using the abbreviations shown in [Table 1-2](#).

Table 1-2 Bit Function Terminology

Function of Bits	Description
Unimplemented	Register bit fields named “0” indicate unimplemented functions with the following behavior. Reading these bit fields returns 0. Writing to these bit fields has no effect. These bit fields are reserved. When writing, software should always set such bit fields to 0 in order to preserve compatibility with future products. Setting the bit fields to 1 may lead to unpredictable results.
Undefined	Certain bit combinations in a bit field can be labeled “Reserved”, indicating that the behavior of the XC82x is undefined for that combination of bits. Setting the register to undefined bit combinations may lead to unpredictable results. Such bit combinations are reserved. When writing, software must always set such bit fields to legal values as provided in the bit field description tables.
rw	The bit or bit field can be read and written.
r	The bit or bit field can only be read (read-only).
w	The bit or bit field can only be written (write-only). Reading always return 0.
h	The bit or bit field can also be modified by hardware (such as a status bit). This attribute can be combined with ‘rw’ or ‘r’ bits to ‘rwh’ and ‘rh’ bits, respectively.

1.7 Acronyms

[Table 1-3](#) lists the acronyms used in this document.

Table 1-3 Acronyms

Acronym	Description
ADC	Analog-to-Digital Converter
ALU	Arithmetic/Logic Unit
BSL	Boot-Loader

Table 1-3 Acronyms (cont'd)

Acronym	Description
CAN	Controller Area Network
CCU6	Capture/Compare Unit 6
CGU	Clock Generation Unit
CORDIC	Cordinate Rotation Digital Computer
CPU	Central Processing Unit
DAP	Device Access Port
DNL	Differential Non-Linearity error
ECC	Error Correction Code
EVR	Embedded Voltage Regulator
FDR	Fractional Divider
FIFO	First-In-First-Out data buffer mechanism
GPIO	General Purpose I/O
IAP	In-Application Programming
IIC	Inter-IC Bus
I/O	Input/Output
INL	Integral Non-Linearity error
ISP	In-System Programming
JTAG	Joint Test Action Group
LEDTSU	LED and Touch-Sense Controller Unit
LIN	Local Interconnect Network
LSB _n	Least Significant Bit: finest granularity of the analog value in digital format, represented by one least significant bit of the conversion result with n bits resolution (measurement range divided in 2 ⁿ equally distributed steps)
MDU	Multiplication/Division Unit
NMI	Non-Maskable Interrupt
OCDS	On-Chip Debug Support
ORC	Out of Range Comparator
PC	Program Counter
POR	Power-On Reset
PLL	Phase-Locked Loop

Table 1-3 Acronyms (cont'd)

Acronym	Description
PSW	Program Status Word
PWM	Pulse Width Modulation
RAM	Random Access Memory
ROM	Read-Only Memory
RTC	Real-Time Clock
SCU	System Control Unit of the device
SFR	Special Function Register
SPD	Single Pin DAP
SPI	Serial Peripheral Interface
SSC	Synchronous Serial Channel
TUE	Total Unadjusted Error
UART	Universal Asynchronous Receiver/Transmitter
WDT	Watchdog Timer

2 XC800 Core

This chapter describes the XC800 Core.

2.1 Overview

The XC800 Core is a complete, high performance CPU core that is functionally upward compatible to the 8051. While the standard 8051 core is designed around a 12-clock machine cycle, the XC800 Core uses a two-clock period machine cycle.

The instruction set consists of 45% one-byte, 41% two-byte and 14% three-byte instructions. Each instruction takes 1, 2 or 4 machine cycles to execute. In case of access to slower memory, the access time may be extended by wait cycles (one wait cycle lasts one machine cycle, which is equivalent to two wait states).

The XC800 Core support a range of debugging features including basic stop/start, single-step execution, breakpoint support and read/write access to the data memory, program memory and special function registers.

Features

The key features of the XC800 Core implemented are listed below.

- Two clocks per machine cycle
- On-chip XRAM in external data space
- 256 bytes IRAM in internal data space
- Up to 64 kByte of code space (not fully assigned to implemented memory)
- Support for synchronous or asynchronous program and data memory
- Wait state support for slow memory
- 15-source, 4-level interrupt controller
- 2 data pointers
- Power saving modes
- Dedicated debug mode
- Two 16-bit timers (Timer 0 and Timer 1)
- Full duplex serial port (UART)

2.2 XC800 Core Functional Blocks

Figure 2-1 shows the functional blocks of the XC800 Core. The XC800 Core consists mainly of the instruction decoder, the arithmetic section, the program control section, the access control section, and the interrupt controller.

The instruction decoder decodes each instruction and accordingly generates the internal signals required to control the functions of the individual units within the core. These internal signals have an effect on the source and destination of data transfers and control the ALU processing.

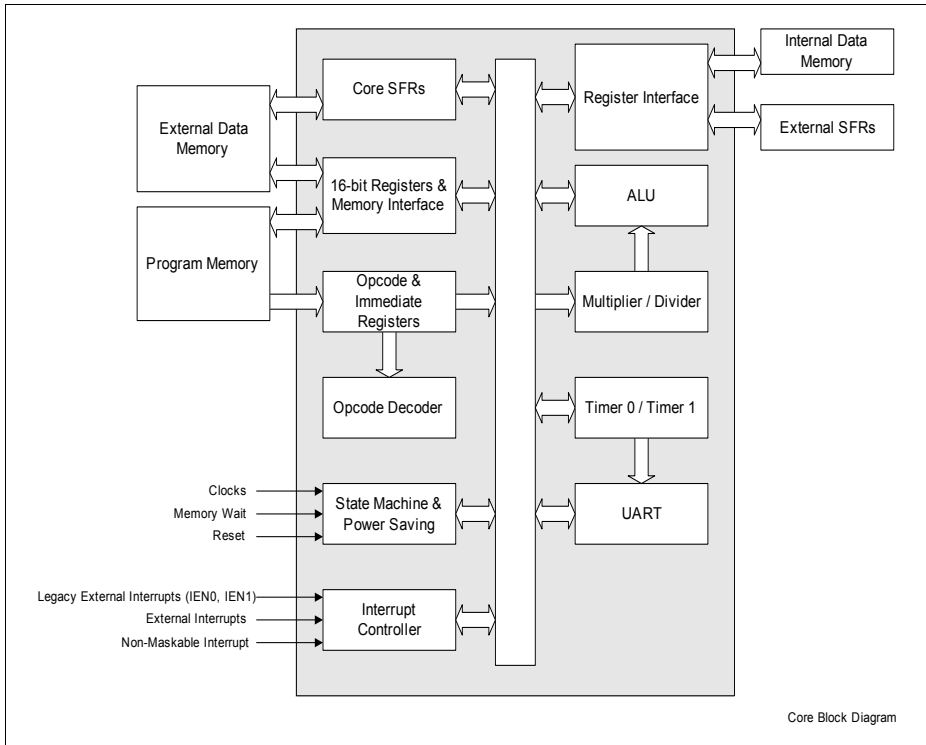


Figure 2-1 XC800 Core Block Diagram

The arithmetic section of the processor performs extensive data manipulation and consists of the arithmetic/logic unit (ALU), A register, B register and PSW register. The ALU accepts 8-bit data words from one or two sources and generates an 8-bit result under the control of the instruction decoder. The ALU performs both arithmetic and logic operations. Arithmetic operations include add, subtract, multiply, divide, increment, decrement, BCD-decimal-add-adjust and compare. Logic operations include AND, OR, Exclusive OR, complement and rotate (right, left or swap nibble (left four)). Also included is a Boolean unit performing the bit operations as set, clear, complement, jump-if-set, jump-if-not-set, jump-if-set-and-clear and move to/from carry. The ALU can perform the bit operations of logical AND or logical OR between any addressable bit (or its complement) and the carry flag, and place the new result in the carry flag.

The program control section controls the sequence in which the instructions stored in program memory are executed. The 16-bit program counter (PC) holds the address of the next instruction to be executed. The conditional branch logic enables internal and external events to the processor to cause a change in the program execution sequence.

XC800 Core

The access control unit is responsible for the selection of the on-chip memory resources. The interrupt requests from the peripheral units are handled by the interrupt controller unit.

2.3 SFRs of the CPU

The XC800 Core registers occupy direct Internal Data Memory space locations in the range 80_H to FF_H.

2.3.1 Stack Pointer (SP, 81_H)

The SP register contains the Stack Pointer. The Stack Pointer is used to load the program counter into internal data memory during LCALL and ACALL instructions and to retrieve the program counter from memory during RET and RETI instructions. Data may also be saved on or retrieved from the stack using PUSH and POP instructions. Instructions that use the stack automatically pre-increment or post-decrement the stack pointer so that the stack pointer always points to the last byte written to the stack, i.e. the top of the stack. On reset, the Stack Pointer is reset to 07_H. This causes the stack to begin at a location = 08_H above register bank zero. The SP can be read or written under software control. The programmer must ensure that the location and size of the stack in internal data memory do not overlap with other application data.

2.3.2 Data Pointer (DPTR, 82-3_H)

The Data Pointer (DPTR) is stored in registers DPL (Data Pointer Low byte) and DPH (Data Pointer High byte) to form 16-bit addresses for External Data Memory accesses (MOVX A,@DPTR and MOVX @DPTR,A), for program byte moves (MOVC A,@A+DPTR) and for indirect program jumps (JMP @A+DPTR).

Two true 16-bit operations are allowed on the Data Pointer: load immediate (MOV DPTR,#data) and increment (INC DPTR).

2.3.3 Accumulator (ACC, E0_H)

This register provides one of the operands for most ALU operations. ACC is the symbol for the accumulator register. The mnemonics for accumulator-specific instructions, however, refer to the accumulator simply as "A".

2.3.4 B Register (F0_H)

The B register is used during multiply and divide operations to provide the second operand. For other instructions, it can be treated as another scratch pad register.

2.3.5 Program Status Word (PSW, D0_H)

The PSW contains several status bits that reflect the current state of the core.

PSW
Program Status Word Register (D0_H)
Reset Value: 00_H
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	F1	P
rwh	rwh	rw	rw	rw	rwh	rw	rh

Field	Bits	Type	Description
P	0	rh	Parity Flag Set/cleared by hardware after each instruction to indicate an odd/even number of “one” bits in the accumulator, i.e. even parity.
F1	1	rw	General Purpose Flag
OV	2	rwh	Overflow Flag Used by arithmetic instructions.
RS0, RS1	3, 4	rw	Register Bank Select These bits are used to select one of the four register banks. 00 _B Bank 0 selected, data address 00 _H - 07 _H 01 _B Bank 1 selected, data address 08 _H - 0F _H 10 _B Bank 2 selected, data address 10 _H - 17 _H 11 _B Bank 3 selected, data address 18 _H - 1F _H
F0	5	rw	General Purpose Flag
AC	6	rwh	Auxiliary Carry Flag Used by instructions which execute BCD operations.
CY	7	rwh	Carry Flag Used by arithmetic instructions.

2.3.6 Extended Operation Register (EO, A2_H)

The instruction set includes an additional instruction **MOVC @(DPTR++),A** which writes to program memory implemented as RAM. This instruction may be used both to download code into the program memory when the CPU is initialized and subsequently, to provide software updates. The instruction copies the contents of the accumulator to the code memory at the location pointed to by the current data pointer, then increments the data pointer.

The instruction uses the opcode A5_H, which is the same as the software break instruction TRAP (see [Table 2-1](#)). Bit TRAP_EN in the Extended Operation (EO) register is used to select the instruction executed by the opcode A5_H. When bit TRAP_EN is 0 (default), the A5_H opcode executes the MOVC instruction. When bit TRAP_EN is 1, the A5_H opcode executes the software break instruction TRAP, which switches the CPU to debug mode for breakpoint processing.

Register EO is also used to select the current data pointer.

EO

Extended Operation Register

(A2_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
	0		TRAP_EN	0	DPSEL2	DPSEL1	DPSEL0
	r		rw	r	rw	rw	rw

Field	Bits	Type	Description
DPSEL0, DPSEL1, DPSEL2	0, 1, 2	rw	Data Pointer Select These bits are used to select the current data pointer. 000 _B DPTR0 selected 001 _B DPTR1 selected others: Reserved
TRAP_EN	4	rw	TRAP Enable 0 _B Select MOVC @(DPTR++),A 1 _B Select software TRAP instruction
0	[7:5]	r	Reserved Returns 0 if read; should be written with 0.

2.3.7 Power Control Register (PCON, 87_H)

The PCON register provides control for entering idle mode, baud rate control for UART in mode 2, as well as two general purpose flags.

The XC800 Core has two power-saving modes: idle mode and power-down mode. The idle mode can be entered via the PCON register. In idle mode, the clock to the core is disabled while the timers, serial port and interrupt controller continue to run. In power-down mode 1, the clock to the entire core is stopped.

PCON

Power Control Register

(87_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
SMOD		0		GF1	GF0	0	IDLE
rw		r		rw	rw	r	rw

Field	Bits	Type	Description
IDLE	0	rw	Idle Mode Enable 0 _B Do not enter idle mode 1 _B Enter idle mode
GF0	2	rw	General Purpose Flag Bit 0
GF1	3	rw	General Purpose Flag Bit 1
SMOD	7	rw	Double Baud Rate Enable This bit controls the baud rate generation for UART in mode 2. 0 _B Do not double the baud rate 1 _B Double the baud rate
0, 0	1, [6:4]	r	Reserved Returns 0 if read; should be written with 0.

2.3.8 Interrupt Registers

One non-maskable and fourteen maskable interrupt nodes are available.

Refer to Interrupt chapter for details of the interrupt registers.

2.4 SFRs of The Core Peripherals

2.4.1 Timer Registers

Two 16-bit timers are provided - Timer 0 (T0) and Timer 1 (T1).

Refer to Timer 0 and Timer 1 chapter for details of the timer registers.

2.4.2 UART Registers

The UART uses three SFRs - PCON, SCON and SBUF.

Refer to [Section 2.3.7](#) and UART chapter for details of the UART registers.

2.5 Instruction Timing

A CPU machine cycle comprises two input clock periods, referred to as Phase 1 (P1) and Phase 2 (P2), that correspond to two different CPU states. A CPU state within an instruction is referenced by the machine cycle and state number, e.g., C2P1 means the first clock period within machine cycle 2. Memory access takes place during one or both phases of the machine cycle. SFR writes occur only at the end of P2. Instructions are 1, 2, or 3 bytes long and can take 1, 2 or 4 machine cycles to execute. Registers are generally updated and the next opcode pre-fetched at the end of P2 of the last machine cycle for the current instruction.

The XC800 Core supports access to slow memory by using wait cycle(s). Each wait cycle lasts one machine cycle, i.e. two clock periods. For example, in case of a memory requiring one/two wait state(s), the access time is increased by one machine cycle for every byte of opcode/operand fetched.

Figure 2-2 shows the fetch/execute timing related to the internal states and phases. Execution of an instruction occurs at C1P1. For a 2-byte instruction, the second reading starts at C1P1.

Figure 2-2 (a) shows two timing diagrams for a 1-byte, 1-cycle ($1 \times$ machine cycle) instruction. The first diagram shows the instruction being executed within one machine cycle since the opcode (C1P2) is fetched from a memory without wait state. The second diagram shows the corresponding states of the same instruction being executed over two machine cycles (instruction time extended), with one wait cycle inserted for opcode fetching from the flash memory.

Figure 2-2 (b) shows two timing diagrams for a 2-byte, 1-cycle ($1 \times$ machine cycle) instruction. The first diagram shows the instruction being executed within one machine cycle since the second byte (C1P1) and the opcode (C1P2) are fetched from a memory without wait state. The second diagram shows the corresponding states of the same instruction being executed over three machine cycles (instruction time extended), with one wait cycle inserted for each access to the flash memory. In this case, two wait cycles are inserted in total.

Figure 2-2 (c) shows two timing diagrams of a 1-byte, 2-cycle ($2 \times$ machine cycle) instruction. The first diagram shows the instruction being executed over two machine cycles with the opcode (C2P2) fetched from a memory without wait state. The second diagram shows the corresponding states of the same instruction being executed over three machine cycles (instruction time extended), with one wait cycle inserted for opcode fetching from the slow memory requiring one/two wait state(s).

Note: For instructions that are executed over two or more machine cycles, execution cycle may or may not be extended in case of access to slow memory with one/two wait states. The execution cycle is, nonetheless, guaranteed consistent for each instruction when accessed from slow memory with defined wait state(s).

Reference: **Table 2-1**.

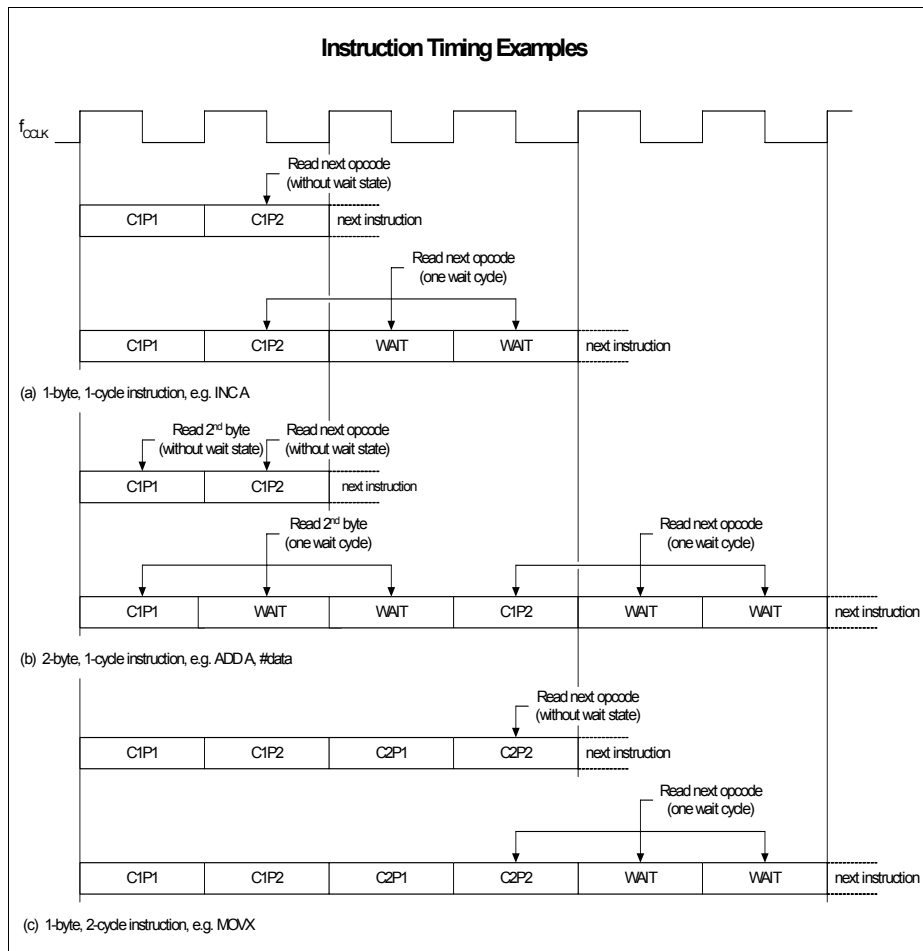


Figure 2-2 CPU Instruction Timing

The time taken for each instruction includes:

- decoding/executing the fetched opcode
- fetching the operand/s (for instructions > 1 byte)
- fetching the first byte (opcode) of the next instruction (due to CPU pipeline)

Note: The XC800 Core fetches the opcode of the next instruction while executing the current instruction.

XC800 Core

Table 2-1 lists all the instructions supported by the XC800 Core. Instructions are 1, 2 or 3 bytes long as indicated in the 'Bytes' column. Each instruction takes 1, 2 or 4 machine cycles to execute (with no wait cycle). The table gives two values for the number of machine cycles required by each instruction. The first value applies to fetching operand/s and opcode from fast memory (e.g. Boot ROM and XRAM) without wait state. The second value applies to fetching operand/s and opcode (and in some cases accessing data) from slow memory (e.g. Flash) with wait cycles inserted due to memory requiring one/two wait state(s). One machine cycle comprises two CCLK clock cycles.

Table 2-1 Instruction Table

Mnemonic	Hex Code	Bytes	Machine Cycles (no wait state)	Machine Cycles (one wait state¹⁾)
ARITHMETIC				
ADD A,Rn	28-2F	1	1	2
ADD A,dir	25	2	1	3
ADD A,@Ri	26-27	1	1	2
ADD A,#data	24	2	1	3
ADDC A,Rn	38-3F	1	1	2
ADDC A,dir	35	2	1	3
ADDC A,@Ri	36-37	1	1	2
ADDC A,#data	34	2	1	3
SUBB A,Rn	98-9F	1	1	2
SUBB A,dir	95	2	1	3
SUBB A,@Ri	96-97	1	1	2
SUBB A,#data	94	2	1	3
INC A	04	1	1	2
INC Rn	08-0F	1	1	2
INC dir	05	2	1	3
INC @Ri	06-07	1	1	2
DEC A	14	1	1	2
DEC Rn	18-1F	1	1	2
DEC dir	15	2	1	3
DEC @Ri	16-17	1	1	2
INC DPTR	A3	1	2	2

Table 2-1 Instruction Table (cont'd)

Mnemonic	Hex Code	Bytes	Machine Cycles (no wait state)	Machine Cycles (one wait state¹⁾)
MUL AB	A4	1	4	4
DIV AB	84	1	4	4
DA A	D4	1	1	2
LOGICAL				
ANL A,Rn	58-5F	1	1	2
ANL A,dir	55	2	1	3
ANL A,@Ri	56-57	1	1	2
ANL A,#data	54	2	1	3
ANL dir,A	52	2	1	3
ANL dir,#data	53	3	2	5
ORL A,Rn	48-4F	1	1	2
ORL A,dir	45	2	1	3
ORL A,@Ri	46-47	1	1	2
ORL A,#data	44	2	1	3
ORL dir,A	42	2	1	3
ORL dir,#data	43	3	2	5
XRL A,Rn	68-6F	1	1	2
XRL A,dir	65	2	1	3
XRL A,@Ri	66-67	1	1	2
XRL A,#data	64	2	1	3
XRL dir,A	62	2	1	3
XRL dir,#data	63	3	2	5
CLR A	E4	1	1	2
CPL A	F4	1	1	2
SWAP A	C4	1	1	2
RL A	23	1	1	2
RLC A	33	1	1	2
RR A	03	1	1	2
RRC A	13	1	1	2

Table 2-1 Instruction Table (cont'd)

Mnemonic	Hex Code	Bytes	Machine Cycles (no wait state)	Machine Cycles (one wait state ¹⁾)
DATA TRANSFER				
MOV A,Rn	E8-EF	1	1	2
MOV A,dir	E5	2	1	3
MOV A,@Ri	E6-E7	1	1	2
MOV A,#data	74	2	1	3
MOV Rn,A	F8-FF	1	1	2
MOV Rn,dir	A8-AF	2	2	4
MOV Rn,#data	78-7F	2	1	3
MOV dir,A	F5	2	1	3
MOV dir,Rn	88-8F	2	2	4
MOV dir,dir	85	3	2	5
MOV dir,@Ri	86-87	2	2	4
MOV dir,#data	75	3	2	5
MOV @Ri,A	F6-F7	1	1	2
MOV @Ri,dir	A6-A7	2	2	4
MOV @Ri,#data	76-77	2	1	3
MOV DPTR,#data	90	3	2	5
MOVC A,@A+DPTR	93	1	2	3 or 4 ²⁾
MOVC A,@A+PC	83	1	2	3 or 4 ²⁾
MOVX A,@Ri	E2-E3	1	2	3
MOVX A,@DPTR	E0	1	2	3
MOVX @Ri,A	F2-F3	1	2	3
MOVX @DPTR,A	F0	1	2	3
PUSH dir	C0	2	2	4
POP dir	D0	2	2	4
XCH A,Rn	C8-CF	1	1	2
XCH A,dir	C5	2	1	3
XCH A,@Ri	C6-C7	1	1	2
XCHD A,@Ri	D6-D7	1	1	2

Table 2-1 Instruction Table (cont'd)

Mnemonic	Hex Code	Bytes	Machine Cycles (no wait state)	Machine Cycles (one wait state ¹⁾)
BOOLEAN				
CLR C	C3	1	1	2
CLR bit	C2	2	1	3
SETB C	D3	1	1	2
SETB bit	D2	2	1	3
CPL C	B3	1	1	2
CPL bit	B2	2	1	3
ANL C,bit	82	2	2	4
ANL C,/bit	B0	2	2	4
ORL C,bit	72	2	2	4
ORL C,/bit	A0	2	2	4
MOV C,bit	A2	2	1	3
MOV bit,C	92	2	2	4
BRANCHING³⁾				
ACALL addr11	11->F1	2	2	4
LCALL addr16	12	3	2	5
RET	22	1	2	2 or 3 ²⁾
RETI	32	1	2	2 or 3 ²⁾
AJMP addr 11	01->E1	2	2	4
LJMP addr 16	02	3	2	5
SJMP rel	80	2	2	4
JC rel	40	2	2	4
JNC rel	50	2	2	4
JB bit,rel	20	3	2	5
JNB bit,rel	30	3	2	5
JBC bit,rel	10	3	2	5
JMP @A+DPTR	73	1	2	2 or 3 ²⁾
JZ rel	60	2	2	4
JNZ rel	70	2	2	4

Table 2-1 Instruction Table (cont'd)

Mnemonic	Hex Code	Bytes	Machine Cycles (no wait state)	Machine Cycles (one wait state ¹⁾)
CJNE A,dir,rel	B5	3	2	5
CJNE A,#d,rel	B4	3	2	5
CJNE Rn,#d,rel	B8-BF	3	2	5
CJNE @Ri,#d,rel	B6-B7	3	2	5
DJNZ Rn,rel	D8-DF	2	2	4
DJNZ dir,rel	D5	3	2	5

MISCELLANEOUS

NOP	00	1	1	2
-----	----	---	---	---

ADDITIONAL INSTRUCTIONS

MOVC @(DPTR++),A	A5	1	2	2 or 3 ²⁾
TRAP	A5	1	1	—

1) In case of fetch from slow memory requiring only 1 wait state, no wait cycle may be required per the instruction fetched (normally for opcodes that do not require fetch in the next cycle).

2) Depending on whether the operation is accessing memory with zero or one/two wait state.

3) For branch instructions, the instruction time may vary depending on jump destination.

3 Memory Organization

The XC82x CPU operates in the following five address spaces:

- 8 Kbytes of Boot ROM program memory
- 256 bytes of internal RAM data memory
- 256 bytes of XRAM memory
(XRAM can be read/written as program memory or external data memory)
- a 128-byte Special Function Register area
- 4 Kbytes of Flash program memory

Figure 3-1 illustrates the memory address spaces of the XC82x.

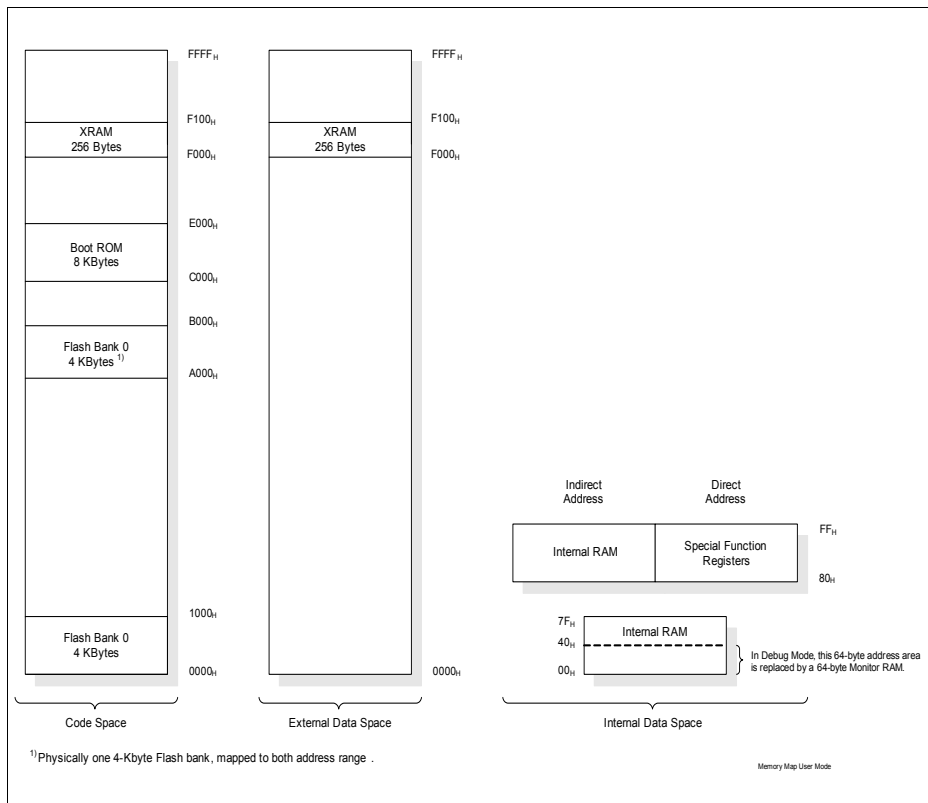


Figure 3-1 Memory Map of XC82x

3.1 Program Memory

The code space is theoretically 64 KBytes. However, only access to defined program memory (as shown in memory map figure) is supported. For XC82x, defined code space is occupied by on-chip memories.

3.2 Data Memory

The data space consists of an internal and external data space. Access to internal and external data space are distinguished by different sets of instruction opcodes. In XC82x, on-chip XRAM is located in external data space and accessed by MOVX instructions. XC82x does not support access to external (off-chip) memory. Internal data space is occupied by Internal RAM (IRAM) and Special Function Registers (SFRs), distinguished by direct or indirect addressing.

3.2.1 Internal Data Memory

The internal data memory is divided into two physically separate and distinct blocks: the 256-byte RAM and the 128-byte Special Function Register (SFR) area. While the upper 128 bytes of RAM and the SFR area share the same address locations, they are accessed through different addressing modes. The lower 128 bytes of RAM can be accessed through either direct or register indirect addressing, while the upper 128 bytes of RAM can be accessed through register indirect addressing only. The SFRs are accessible through direct addressing.

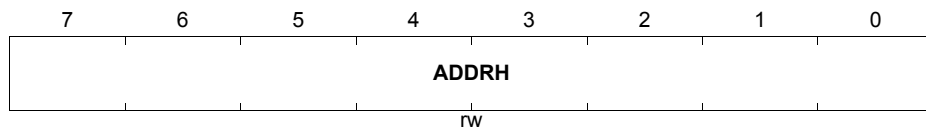
The 16 bytes of RAM that occupy addresses from 20_H to $2F_H$ are bitaddressable. RAM occupying direct addresses from 30_H to $7F_H$ can be used as scratch pad registers or used for the stack.

3.2.2 External Data Memory

The 256-byte XRAM is mapped to both the external data memory area and the program memory area. It can be accessed using both 'MOVX' and 'MOVC' instructions.

The 'MOVX' instructions for XRAM access use either 8-bit or 16-bit indirect addresses. While the DPTR register is used for 16-bit addressing, either register R0 or R1 is used to form the 8-bit address. The upper byte of the XRAM address during execution of the 8-bit accesses is defined by the value stored in register XADDRH. Hence, the write instruction for setting the higher order XRAM address in register XADDRH must precede the 'MOVX' instruction.

The bit field PAGE of SCU_PAGE register must be programmed before accessing the XADDRH register.

Memory Organization
XADDRH
On-Chip XRAM Address Higher Order (F2_H)
Reset Value: F0_H
RMAP: 0, PAGE: 3


Field	Bits	Type	Description
ADDRH	[7:0]	rw	Higher Order of On-chip XRAM Address The value for XC82x is F0 _H .

3.3 Memory Protection Strategy

The memory protection strategy in XC82x prevents unauthorized read out of critical data and user IP from Flash memory by blocking all external access to the device.

This is achieved by using the Boot Mode Index (BMI) to control the boot options such that once the BMI is programmed to enter user mode (productive), it is not allowed to enter the other boot modes without an erase of the BMI by the user code.

Therefore, boot options that load and execute external code will be blocked and only user code starting from address 0000_H can be executed.

3.4 Special Function Registers

The Special Function Registers (SFRs) occupy direct internal data memory space in the range 80_H to FF_H. All registers, except the program counter, reside in the SFR area. The SFRs include pointers and registers that provide an interface between the CPU and the on-chip peripherals. As the 128-SFR range is less than the total number of registers required, address extension mechanisms are required to increase the number of addressable SFRs. The address extension mechanisms include:

- Mapping
- Paging

3.4.1 Address Extension by Mapping

Address extension is performed at the system level by mapping. The SFR area is extended into two portions: the standard (non-mapped) SFR area and the mapped SFR area. Each portion supports the same address range 80_H to FF_H, bringing the number of addressable SFRs to 256. The extended address range is not directly controlled by the CPU instruction itself, but is derived from bit RMAP in the system control register SYSCON0 at address 8F_H. To access SFRs in the mapped area, bit RMAP in SFR SYSCON0 must be set. However, the SFRs in the standard area can be accessed by clearing bit RMAP. **Figure 3-2** shows how the SFR area can be selected.

As long as bit RMAP is set, the mapped SFR area can be accessed. This bit is not cleared automatically by hardware. Thus, before standard/mapped registers are accessed, bit RMAP must be cleared/set, respectively, by software.

Memory Organization

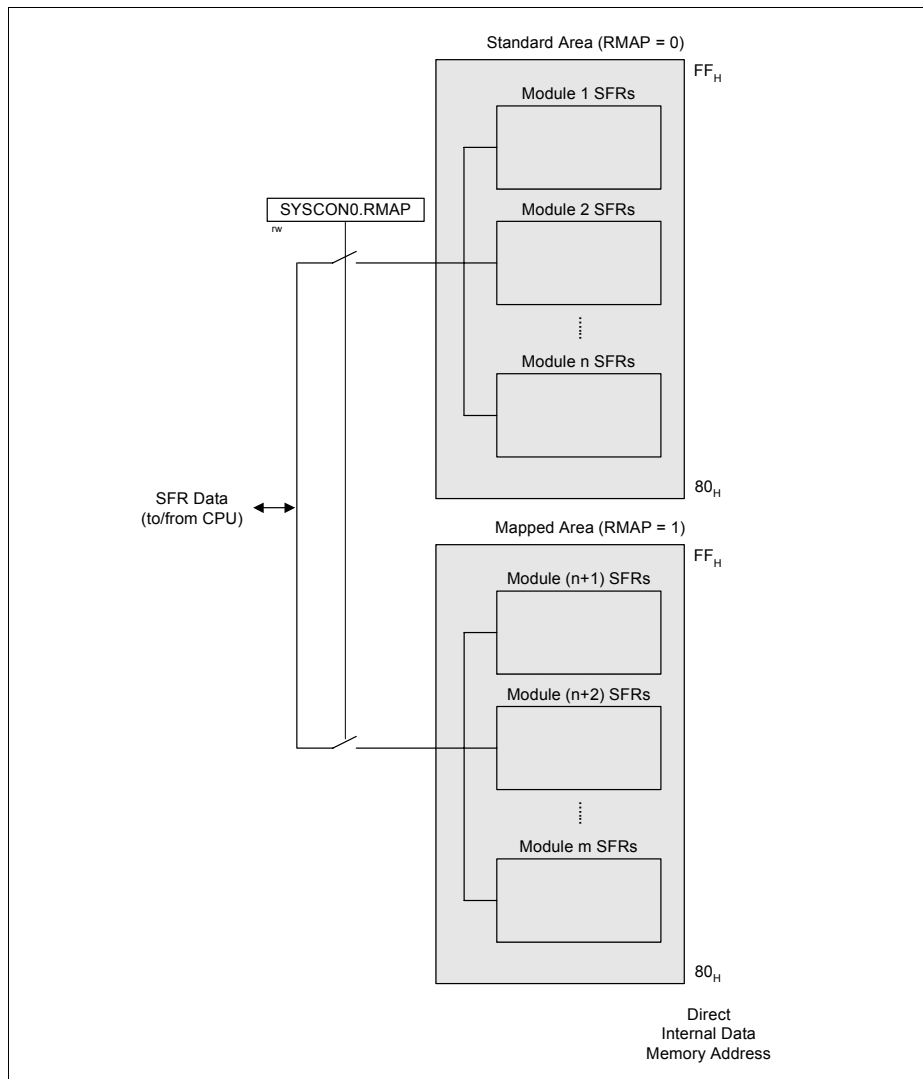


Figure 3-2 Address Extension by Mapping

Memory Organization

3.4.1.1 System Control Register 0

The SYSCON0 register contains the bit to select the SFR mapping.

SYSCON0

System control Register 0

(8F_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
0							RMAP
r							rw

Field	Bits	Type	Description
RMAP	0	rw	Special Function Register Map Control 0 Accessed to non-mapped (standard) special function register area. 1 Accessed to mapped special function register area.
0	[7:1]	r	Reserved Returns 0 if read; should be written with 0.

3.4.2 Address Extension by Paging

Address extension is further performed at the module level by paging. With the address extension by mapping, the XC82x has a 256-SFR address range. However, this is still less than the total number of SFRs needed by the on-chip peripherals. To meet this requirement, some peripherals have a built-in local address extension mechanism for increasing the number of addressable SFRs. The extended address range is not directly controlled by the CPU instruction itself, but is derived from bit field PAGE in the module page register MOD_PAGE. Hence, the bit field PAGE must be programmed before accessing the SFRs of the target module. Each module may contain a different number of pages and a different number of SFRs per page, depending on the specific requirement. Besides setting the correct RMAP bit value to select the SFR area, the user must also ensure that a valid PAGE is selected to target the desired SFRs. **Figure 3-3** shows how a page inside the extended address range can be selected.

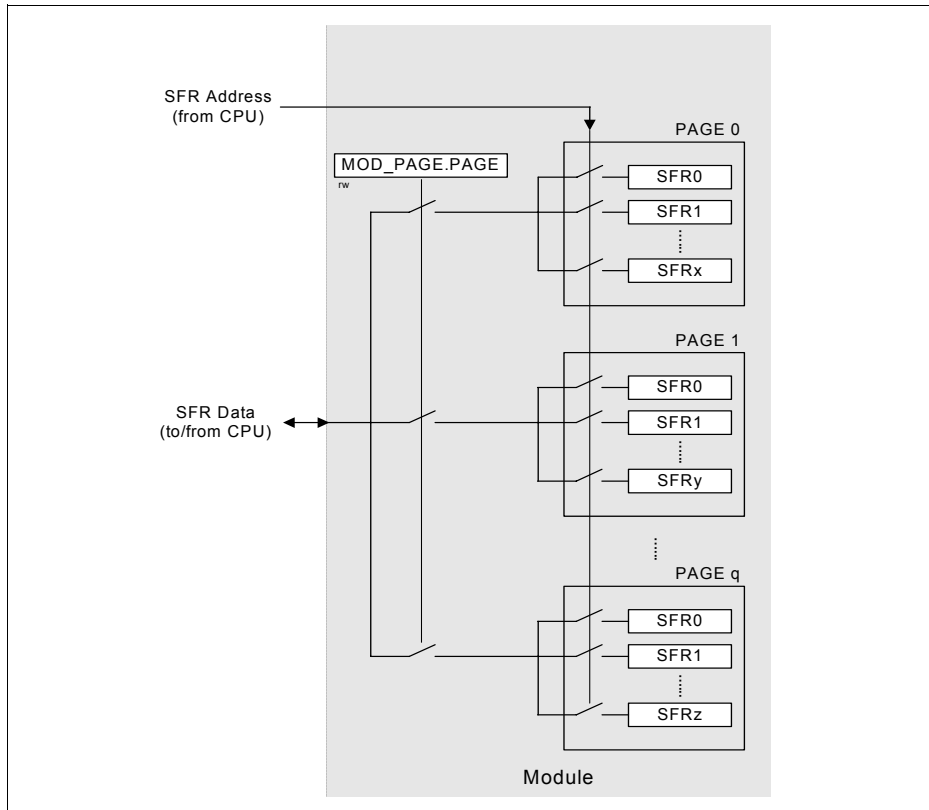


Figure 3-3 Address Extension by Paging

Memory Organization

In order to access a register located in a page other than the current one, the current page must be exited. This is done by reprogramming the bit field PAGE in the page register. Only then can the desired access be performed.

If an interrupt routine is initiated between the page register access and the module register access, and the interrupt needs to access a register located in another page, the current page setting can be saved, the new one programmed, and the old page setting restored. This is possible with the storage fields STx (x = 0 - 3) for the save and restore action of the current page setting. By indicating which storage bit field should be used in parallel with the new page value, a single write operation can:

- Save the contents of PAGE in STx before overwriting with the new value (this is done at the beginning of the interrupt routine to save the current page setting and program the new page number); or
- Overwrite the contents of PAGE with the contents of STx, ignoring the value written to the bit positions of PAGE (this is done at the end of the interrupt routine to restore the previous page setting before the interrupt occurred)

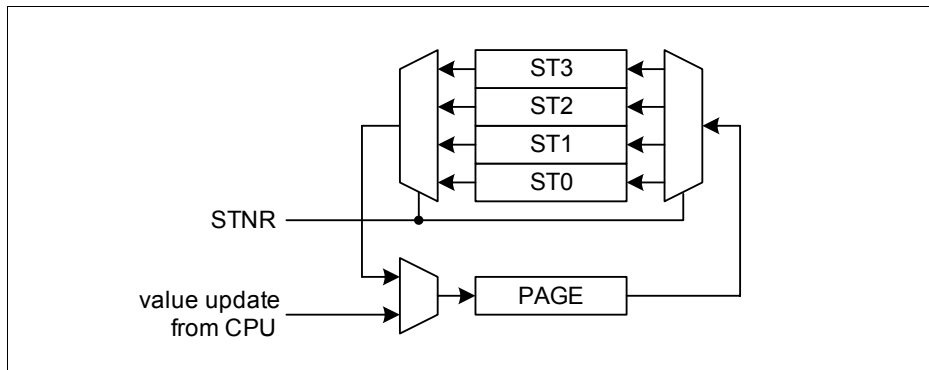


Figure 3-4 Storage Elements for Paging

With this mechanism, a certain number of interrupt routines (or other routines) can perform page changes without reading and storing the previously used page information. The use of only write operations makes the system simpler and faster. Consequently, this mechanism significantly improves the performance of short interrupt routines.

The XC82x supports local address extension for:

- Parallel Ports
- Analog-to-Digital Converter (ADC)
- Capture/Compare Unit 6 (CCU6)
- System Control Registers

Memory Organization

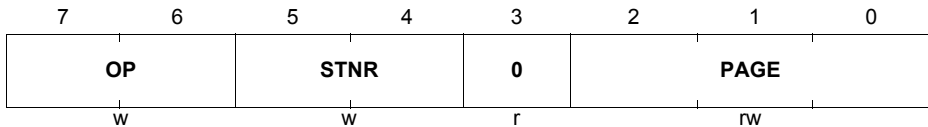
3.4.2.1 Page Register

The page register has the following definition:

MOD_PAGE

Page Register for module MOD

Reset Value: 00_H



Field	Bits	Type	Description
PAGE	[2:0]	rw	Page Bits When written, the value indicates the new page. When read, the value indicates the currently active page.
STNR	[5:4]	w	Storage Number This number indicates which storage bit field is the target of the operation defined by bit field OP. If OP = 10 _B , the contents of PAGE are saved in STx before being overwritten with the new value. If OP = 11 _B , the contents of PAGE are overwritten by the contents of STx. The value written to the bit positions of PAGE is ignored. 00 ST0 is selected. 01 ST1 is selected. 10 ST2 is selected. 11 ST3 is selected.

Memory Organization

Field	Bits	Type	Description
OP	[7:6]	w	Operation 0X Manual page mode. The value of STNR is ignored and PAGE is directly written. 10 New page programming with automatic page saving. The value written to the bit positions of PAGE is stored. In parallel, the previous contents of PAGE are saved in the storage bit field STx indicated by STNR. 11 Automatic restore page action. The value written to the bit positions PAGE is ignored and instead, PAGE is overwritten by the contents of the storage bit field STx indicated by STNR.
0	3	r	Reserved Returns 0 if read; should be written with 0.

3.4.3 Bit-Addressing

SFRs that have addresses in the form of 1XXXX000_B (e.g., 80_H, 88_H, 90_H, ..., F0_H, F8_H) are bitaddressable.

3.4.4 Bit Protection Scheme

The bit protection scheme prevents direct software writing of selected bits (i.e., protected bits) using the PASSWD register. When the bit field MODE is 11_B, writing 10011_B to the bit field PASS opens access to writing of all protected bits, and writing 10101_B to the bit field PASS closes access to writing of all protected bits. In both cases, the value of the bit field MODE is not changed even if PASSWD register is written with 98_H or A8_H. It can only be changed when bit field PASS is written with 11000_B, for example, writing D0_H to PASSWD register disables the bit protection scheme.

Note that access is opened for maximum 32 CCLKs if the “close access” password is not written. If “open access” password is written again before the end of 32 CCLK cycles, there will be a recount of 32 CCLK cycles. The protected bits are: include the soft reset request bit, SWRQ; the Watchdog Timer enable bit, WDTEN; the RTC clock count registers, CNT0 - 5; and the power-down enable bit, PD.

- RSTCON.SWRQ: soft reset request bit
- PMCON0.PD: power-down enable bit
- WDTCON.WDTEN: WDT enable bit
- RTC_CNTx (x = 0 - 5): all bits of RTC clock count registers

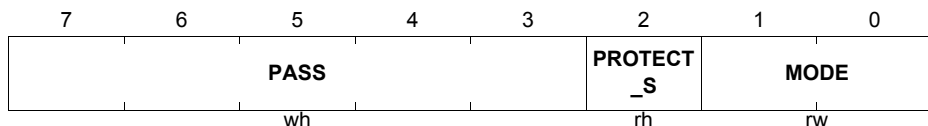
The bit field PAGE of SCU_PAGE register must be programmed before accessing the PASSWD register.

PASSWD

Password Register

Reset Value: 07_H

RMAP: 0, PAGE: 1



Memory Organization

Field	Bits	Type	Description
MODE	[1:0]	rw	Bit Protection Scheme Control bits 00 Scheme disabled - direct access to the protected bits is allowed. 11 Scheme enabled - the bit field PASS has to be written with the passwords to open and close the access to protected bits. (default) Others: Scheme enabled These two bits cannot be written directly. To change the value between 11 _B and 00 _B , the bit field PASS must be written with 11000 _B ; only then, will the MODE[1:0] be registered.
PROTECT_S	2	rh	Bit Protection Signal Status bit This bit shows the status of the protection. 0 Software is able to write to all protected bits. 1 Software is unable to write to any protected bits.
PASS	[7:3]	wh	Password bits The Bit Protection Scheme only recognizes three patterns. 11000 _B Enables writing of the bit field MODE. 10011 _B Opens access to writing of all protected bits. 10101 _B Closes access to writing of all protected bits.

Memory Organization

3.4.5 XC82x Register Overview

The SFRs of the XC82x are organized into groups according to their functional units. The contents (bits) of the SFRs are summarized in [Section 3.4.5.1](#) to [Section 3.4.5.12](#).

Note: The addresses of the bit addressable SFRs appear in bold typeface.

3.4.5.1 CPU Registers

The CPU SFRs can be accessed in both the standard and mapped memory areas (RMAP = 0 or 1).

Table 3-1 CPU Register Overview

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
RMAP = 0 or 1										
81 _H	SP Stack Pointer Register Reset: 07_H	Bit Field	SP							
		Type	rw							
82 _H	DPL Data Pointer Register Low Reset: 00_H	Bit Field	DPL7	DPL6	DPL5	DPL4	DPL3	DPL2	DPL1	DPL0
		Type	rw	rw	rw	rw	rw	rw	rw	rw
83 _H	DPH Data Pointer Register High Reset: 00_H	Bit Field	DPH7	DPH6	DPH5	DPH4	DPH3	DPH2	DPH1	DPH0
		Type	rw	rw	rw	rw	rw	rw	rw	rw
87 _H	PCON Power Control Register Reset: 00_H	Bit Field	SMOD	0			GF1	GF0	0	IDLE
		Type	rw	r			rw	rw	r	rw
88 _H	TCON Timer Control Register Reset: 00_H	Bit Field	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
		Type	rwh	rw	rwh	rw	rwh	rw	rwh	rw
89 _H	TMOD Timer Mode Register Reset: 00_H	Bit Field	GATE 1	T1S	T1M		GATE 0	T0S	T0M	
		Type	rw	rw	rw		rw	rw	rw	
8A _H	TL0 Timer 0 Register Low Reset: 00_H	Bit Field	VAL							
		Type	rwh							
8B _H	TL1 Timer 1 Register Low Reset: 00_H	Bit Field	VAL							
		Type	rwh							
8C _H	TH0 Timer 0 Register High Reset: 00_H	Bit Field	VAL							
		Type	rwh							
8D _H	TH1 Timer 1 Register High Reset: 00_H	Bit Field	VAL							
		Type	rwh							
98 _H	SCON Serial Channel Control Register Reset: 00_H	Bit Field	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
		Type	rw	rw	rw	rw	rw	rwh	rwh	rwh
99 _H	SBUF Serial Data Buffer Register Reset: 00_H	Bit Field	VAL							
		Type	rwh							
A2 _H	EO Extended Operation Register Reset: 00_H	Bit Field	0			TRAP EN	0	DPSE L2	DPSE L1	DPSE L0
		Type	r			rw	r	rw	rw	rw

Memory Organization
Table 3-1 CPU Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
A8 _H	IEN0 Reset: 00_H Interrupt Enable Register 0	Bit Field	EA	0	ET2	ES	ET1	EX1	ET0	EX0
		Type	rw	r	rw	rw	rw	rw	rw	rw
B8 _H	IP Reset: 00_H Interrupt Priority Register	Bit Field	0		PT2	PS	PT1	PX1	PT0	PX0
		Type	r		rw	rw	rw	rw	rw	rw
B9 _H	IPH Reset: 00_H Interrupt Priority High Register	Bit Field	0		PT2H	PSH	PT1H	PX1H	PT0H	PX0H
		Type	r		rw	rw	rw	rw	rw	rw
D0 _H	PSW Reset: 00_H Program Status Word Register	Bit Field	CY	AC	F0	RS1	RS0	OV	F1	P
		Type	rwh	rwh	rw	rw	rw	rwh	rw	rh
E0 _H	ACC Reset: 00_H Accumulator Register	Bit Field	ACC7	ACC6	ACC5	ACC4	ACC3	ACC2	ACC1	ACC0
		Type	rw	rw	rw	rw	rw	rw	rw	rw
E8 _H	IEN1 Reset: 00_H Interrupt Enable Register 1	Bit Field	ECCIP 3	ECCIP 2	ECCIP 1	ECCIP 0	EXM	EX2	ESSC	EADC
		Type	rw	rw	rw	rw	rw	rw	rw	rw
F0 _H	B Reset: 00_H B Register	Bit Field	B7	B6	B5	B4	B3	B2	B1	B0
		Type	rw	rw	rw	rw	rw	rw	rw	rw
F8 _H	IP1 Reset: 00_H Interrupt Priority 1 Register	Bit Field	PCCIP 3	PCCIP 2	PCCIP 1	PCCIP 0	PXM	PX2	PSSC	PADC
		Type	rw	rw	rw	rw	rw	rw	rw	rw
F9 _H	IPH1 Reset: 00_H Interrupt Priority 1 High Register	Bit Field	PCCIP 3H	PCCIP 2H	PCCIP 1H	PCCIP 0H	PXMH	PX2H	PSSC H	PADC H
		Type	rw	rw	rw	rw	rw	rw	rw	rw

3.4.5.2 MDU Registers

The MDU SFRs can be accessed in the standard memory area (RMAP = 0).

Table 3-2 MDU Register Overview

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
RMAP = 0										
B0 _H	MDUSTAT Reset: 00_H MDU Status Register	Bit Field	0					BSY	IERR	IRDY
		Type	r					rh	rwh	rwh
B1 _H	MDUCON Reset: 00_H MDU Control Register	Bit Field	IE	IR	RSEL	STAR T	OPCODE			
		Type	rw	rw	rw	rwh	rw			
B2 _H	MD0 Reset: 00_H MDU Operand Register 0	Bit Field	DATA							
		Type	rw							
B2 _H	MR0 Reset: 00_H MDU Result Register 0	Bit Field	DATA							
		Type	rh							
B3 _H	MD1 Reset: 00_H MDU Operand Register 1	Bit Field	DATA							
		Type	rw							

Memory Organization

Table 3-2 MDU Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
B3 _H	MR1 Reset: 00_H MDU Result Register 1	Bit Field	DATA							
		Type	rh							
B4 _H	MD2 Reset: 00_H MDU Operand Register 2	Bit Field	DATA							
		Type	rw							
B4 _H	MR2 Reset: 00_H MDU Result Register 2	Bit Field	DATA							
		Type	rh							
B5 _H	MD3 Reset: 00_H MDU Operand Register 3	Bit Field	DATA							
		Type	rw							
B5 _H	MR3 Reset: 00_H MDU Result Register 3	Bit Field	DATA							
		Type	rh							
B6 _H	MD4 Reset: 00_H MDU Operand Register 4	Bit Field	DATA							
		Type	rw							
B6 _H	MR4 Reset: 00_H MDU Result Register 4	Bit Field	DATA							
		Type	rh							
B7 _H	MD5 Reset: 00_H MDU Operand Register 5	Bit Field	DATA							
		Type	rw							
B7 _H	MR5 Reset: 00_H MDU Result Register 5	Bit Field	DATA							
		Type	rh							

3.4.5.3 System Control Registers

The system control SFRs can be accessed in the standard memory area (RMAP = 0).

Table 3-3 SCU Register Overview

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
RMAP = 0 or 1										
8F _H	SYSCON0 Reset: 04_H System Control Register 0	Bit Field	0			IMOD E	0	1	0	RMAP
		Type	r			rw	r	r	r	rw
RMAP = 0										
F1 _H	SCU_PAGE Reset: 00_H Page Register	Bit Field	OP		STNR		0	PAGE		
		Type	w		w		r	rw		
RMAP = 0, PAGE 0										
EE _H	NMICON Reset: 00_H NMI Control Register	Bit Field	0	NMI ECC	NMI VDDP	NMI VDDC	NMI OCDS	NMI FLASH	NMI OSC CLK	NMI WDT
		Type	r	rw	rw	rw	rw	rw	rw	rw
EF _H	EXICON0 Reset: F0_H External Interrupt Control Register 0	Bit Field	EXINT3			EXINT2		EXINT1		EXINT0
		Type	rw			rw		rw		rw

Memory Organization
Table 3-3 SCU Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
F2 _H	IRCON0 Reset: 00_H Interrupt Request Register 0	Bit Field	0	EXINT 6	EXINT 5	EXINT 4	EXINT 3	EXINT 2	0	
		Type	r	rwh	rwh	rwh	rwh	rwh	r	
F3 _H	IRCON1 Reset: 00_H Interrupt Request Register 1	Bit Field	0			ADCS R1	ADCS R0	RIR	TIR	EIR
		Type	r			rwh	rwh	rwh	rwh	rwh
F4 _H	EXICON1 Reset: 3F_H External Interrupt Control Register 1	Bit Field	0		EXINT6		EXINT5		EXINT4	
		Type	r		rw		rw		rw	
F5 _H	IRCON2 Reset: 00_H Interrupt Request Register 2	Bit Field	0			CCU6 SR1	0			CCU6 SR0
		Type	r			rwh	r			rwh
F6 _H	IRCON3 Reset: 00_H Interrupt Request Register 3	Bit Field	0			CCU6 SR3	0			CCU6 SR2
		Type	r			rwh	r			rwh
F7 _H	NMISR Reset: 00_H NMI Status Register	Bit Field	0	FNMI ECC	FNMI VDDP	FNMI VDDC	FNMI OCDS	FNMI FLASH	FNMI OSC CLK	FNMI WDT
		Type	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh
RMAP = 0, PAGE 1										
EE _H	SDCON Reset: 34_H Supply Detection Control Register	Bit Field	0		VDDP TH	VDDC TH	VDDP BOBP	VDDP BOA	VDDP PW	VDDC PW
		Type	r		rh	rh	rw	rw	rw	rw
EF _H	PMCON1 Reset: 00_H Power Mode Control Register 1	Bit Field	IIC_ DIS	LTS_ DIS	0	MDU_ DIS	T2_ DIS	CCU_ DIS	SSC_ DIS	ADC_ DIS
		Type	rw	rw	r	rw	rw	rw	rw	rw
F2 _H	PASSWD Reset: 07_H Password Register	Bit Field	PASS					PROT ECT_ S	MODE	
		Type	wh					rh	rw	
F3 _H	PMCON0 Reset: 01_H Power Mode Control Register 0	Bit Field	0			WK SEL	0	PD MODE	PD	EWS
		Type	r			rw	r	rw	rwh	rw
F4 _H	OSC_CON Reset: 00_H OSC Control Register	Bit Field	0	INT OSC_ ST	0			75K OSC 2L	48M OSC 2L	RC OWD RST
		Type	r	rh	r			rh	rh	rwh
F5 _H	ID Reset: UU_H Identity Register	Bit Field	PRODID					VERID		
		Type	r					r		
F6 _H	WDTCON Reset: 00_H Watchdog Timer Control Register	Bit Field	0		WINB EN	WDT PR	0	WDT EN	WDT RS	0
		Type	r		rw	rh	r	rw	rwh	r
F7 _H	RSTCON Reset: 00_H Reset Control Register	Bit Field	SWRQ	0				SOFT RS	WDT RST	WKRS
		Type	rwh	r				rwh	rwh	rwh
RMAP = 0, PAGE 3										

Memory Organization

Table 3-3 SCU Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
EE _H	MODPISEL3 Reset: 00 _H Peripheral Input Select Register 3	Bit Field	IST13HR1			IST12HR1			CTRAPIS	
		Type	rw			rw			rw	
F2 _H	XADDRH Reset: F0 _H On-chip XRAM Address Higher Order	Bit Field	ADDRH							
		Type	rw							
F3 _H	MODPISEL Reset: 00 _H Peripheral Input Select Register	Bit Field	0	CIS	SIS			0	MIS	
		Type	r	rw	rw			r	rw	
F4 _H	MODPISEL1 Reset: 00 _H Peripheral Input Select Register 1	Bit Field	0	EXINT1IS	0	EXINT0IS		0	URRIS	
		Type	r	rw	r	rw		r	rw	
F5 _H	MODPISEL2 Reset: 00 _H Peripheral Input Select Register 2	Bit Field	0	T0IS	T1IS	T2IS		T2EXIS		
		Type	r	rw	rw	rw		rw		
F6 _H	MODSUSP Reset: 01 _H Module Suspend Control Register	Bit Field	0		LTS SUSP	RTC SUSP	T2SUSP	T13SUSP	T12SUSP	WDT SUSP
		Type	r		rw	rw	rw	rw	rw	rw
F7 _H	MODIEN Reset: 07 _H Peripheral Interrupt Enable Register	Bit Field	CCU6 SR3 EN	CCU6 SR2 EN	0			RIREN	TIREN	EIREN
		Type	rw	rw	r			rw	rw	rw
RMAP = 0, PAGE 4										
F3 _H	WDTREL Reset: 00 _H Watchdog Timer Reload Register	Bit Field	WDTREL							
		Type	rw							
F4 _H	WDTWINB Reset: 00 _H Watchdog Window-Boundary Count Register	Bit Field	WDTWINB							
		Type	rw							
F5 _H	WDTL Reset: 00 _H Watchdog Timer Register Low	Bit Field	WDT							
		Type	rh							
F6 _H	WDTH Reset: 00 _H Watchdog Timer Register High	Bit Field	WDT							
		Type	rh							
RMAP = 0, PAGE 5										
F2 _H	BCON Reset: 00 _H Baud Rate Control Register	Bit Field	BGSEL		0	BRDIS	BRPRE			R
		Type	rw		r	rw	rw			rw
F3 _H	BGL Reset: 00 _H Baud Rate Timer/Reload Register, Low Byte	Bit Field	BR_VALUE			FDSEL				
		Type	rwh			rw				
F4 _H	BGH Reset: 00 _H Baud Rate Timer/Reload Register, High Byte	Bit Field	BR_VALUE							
		Type	rwh							
F5 _H	LINST Reset: 00 _H LIN Status Register	Bit Field	BGS	SYNE N	ERRS YN	EOFS YN	BRK	0		
		Type	rw	rw	rwh	rwh	rwh	r		
F6 _H	FEAL Reset: 00 _H Flash Error Address Register Low	Bit Field	ECCERRADDR							
		Type	rh							

Memory Organization

Table 3-3 SCU Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
7 _H	FEAH Reset: 00_H Flash Error Address Register High	Bit Field	ECCERRADDR							
		Type	rh							

3.4.5.4 Port Registers

The Port SFRs can be accessed in the standard memory area (RMAP = 0).

Table 3-4 Port Register Overview

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
RMAP = 0										
8E _H	PORT_PAGE Reset: 00_H Page Register	Bit Field	OP		STNR		0	PAGE		
		Type	w		w		r	rw		
RMAP = 0, PAGE 0										
80 _H	P0_DATAOUT Reset: 7F_H P0 Data Output Register	Bit Field	0	P6	P5	P4	P3	P2	P1	P0
		Type	r	rw	rw	rw	rw	rw	rw	rw
86 _H	P0_DATAIN Reset: UU_H P0 Data In Register	Bit Field	0	P6	P5	P4	P3	P2	P1	P0
		Type	r	rh	rh	rh	rh	rh	rh	rh
90 _H	P1_DATA Reset: 3F_H P1 Data Register	Bit Field	0		P5	P4	P3	P2	P1	P0
		Type	r		rw	rw	rw	rw	rw	rw
91 _H	P1_DATAIN Reset: UU_H P1 Data In Register	Bit Field	0		P5	P4	P3	P2	P1	P0
		Type	r		rh	rh	rh	rh	rh	rh
94 _H	P2_DATAIN Reset: 0U_H P2 Data In Register	Bit Field	0				P3	P2	P1	P0
		Type	r				rh	rh	rh	rh
RMAP = 0, PAGE 1										
80 _H	P0_PUDEL Reset: 6F_H P0 Pull-Up/Pull-Down Select Register	Bit Field	0	P6	P5	P4	P3	P2	P1	P0
		Type	r	rw	rw	rw	rw	rw	rw	rw
86 _H	P0_PUDEN Reset: C4_H P0 Pull-Up/Pull-Down Enable Register	Bit Field	P7	P6	P5	P4	P3	P2	P1	P0
		Type	rw	rw	rw	rw	rw	rw	rw	rw
90 _H	P1_PUDEL Reset: 3F_H P1 Pull-Up/Pull-Down Select Register	Bit Field	0		P5	P4	P3	P2	P1	P0
		Type	r		rw	rw	rw	rw	rw	rw
91 _H	P1_PUDEN Reset: 00_H P1 Pull-Up/Pull-Down Enable Register	Bit Field	0		P5	P4	P3	P2	P1	P0
		Type	r		rw	rw	rw	rw	rw	rw
93 _H	P2_PUDEL Reset: 0F_H P2 Pull-Up/Pull-Down Select Register	Bit Field	0				P3	P2	P1	P0
		Type	r				rw	rw	rw	rw
94 _H	P2_PUDEN Reset: 00_H P2 Pull-Up/Pull-Down Enable Register	Bit Field	0				P3	P2	P1	P0
		Type	r				rw	rw	rw	rw
RMAP = 0, PAGE 2										

Memory Organization
Table 3-4 Port Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
80 _H	P0_ALTSEL0 Reset: 00_H P0 Alternate Select 0 Register	Bit Field	0	P6	P5	P4	P3	P2	P1	P0
		Type	r	rw	rw	rw	rw	rw	rw	rw
85 _H	P0_ALTSEL2 Reset: 00_H P0 Alternate Select 2 Register	Bit Field	0	P6	P5	P4	0			
		Type	r	rw	rw	rw	r			
86 _H	P0_ALTSEL1 Reset: 00_H P0 Alternate Select 1 Register	Bit Field	0	P6	P5	P4	P3	P2	P1	P0
		Type	r	rw	rw	rw	rw	rw	rw	rw
90 _H	P1_ALTSEL0 Reset: 00_H P1 Alternate Select 0 Register	Bit Field	0		P5	P4	P3	P2	P1	P0
		Type	r		rw	rw	rw	rw	rw	rw
91 _H	P1_ALTSEL1 Reset: 00_H P1 Alternate Select 1 Register	Bit Field	0		P5	P4	P3	P2	P1	P0
		Type	r		rw	rw	rw	rw	rw	rw
RMAP = 0, PAGE 3										
80 _H	P0_OD Reset: 7F_H P0 Open Drain Control Register	Bit Field	0	P6	P5	P4	P3	P2	P1	P0
		Type	r	rw	rw	rw	rw	rw	rw	rw
90 _H	P1_OD Reset: 3F_H P1 Open Drain Control Register	Bit Field	0		P5	P4	P3	P2	P1	P0
		Type	r		rw	rw	rw	rw	rw	rw
94 _H	P2_EN Reset: 00_H P2 Enable Register	Bit Field	0				P3	P2	P1	P0
		Type	r				rw	rw	rw	rw

3.4.5.5 ADC Registers

The ADC SFRs can be accessed in the standard memory area (RMAP = 0).

Table 3-5 ADC Register Overview

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
RMAP = 0										
D1 _H	ADC_PAGE Reset: 00_H Page Register	Bit Field	OP		STNR		0	PAGE		
		Type	w		w		r	rw		
RMAP = 0, PAGE 0										
CA _H	ADC_GLOBCTR Reset: 30_H Global Control Register	Bit Field	ANON	DW	CTC		ORCIE N	CLCIE N	0	
		Type	rw	rw	rw		rw	rw	r	
CB _H	ADC_GLOBSTR Reset: 00_H Global Status Register	Bit Field	0			CHNR		0	SAMP LE	BUSY
		Type	r			rh		r	rh	rh
CC _H	ADC_PRAR Reset: 00_H Priority and Arbitration Register	Bit Field	ASEN 1	ASEN 0	0	ARBM	CSM1	PRI01	CSM0	PRI00
		Type	rw	rw	r	rw	rw	rw	rw	rw
CD _H	ADC_LCBR0 Reset: 70_H Limit Check Boundary Register 0	Bit Field	BOUND0							
		Type	rw							

Memory Organization
Table 3-5 ADC Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
CE _H	ADC_INPCR0 Reset: 00_H Input Class 0 Register	Bit Field	0				STC			
		Type	r				rw			
CF _H	ADC_LCBR1 Reset: B0_H Limit Check Boundary Register 1	Bit Field	BOUND1							
		Type	rw							
D2 _H	ADC_LORE Reset: 00_H Latched Out of Range Event Register	Bit Field	0				LORE 3	LORE 2	LORE 1	LORE 0
		Type	r				rw	rw	rw	rw
D3 _H	ADC_ENORC Reset: 00_H Enable Out of Range Comparator Register	Bit Field	0				ENOR C3	ENOR C2	ENOR C1	ENOR C0
		Type	r				rw	rw	rw	rw
RMAP = 0, PAGE 1										
CA _H	ADC_CHCTR0 Reset: 00_H Channel Control Register 0	Bit Field	BFEN	LCC			REFSEL		RESRSEL	
		Type	rw	rw			rw		rw	
CB _H	ADC_CHCTR1 Reset: 00_H Channel Control Register 1	Bit Field	BFEN	LCC			REFSEL		RESRSEL	
		Type	rw	rw			rw		rw	
CC _H	ADC_CHCTR2 Reset: 00_H Channel Control Register 2	Bit Field	BFEN	LCC			REFSEL		RESRSEL	
		Type	rw	rw			rw		rw	
CD _H	ADC_CHCTR3 Reset: 00_H Channel Control Register 3	Bit Field	0	LCC			REFSEL		RESRSEL	
		Type	r	rw			rw		rw	
RMAP = 0, PAGE 2										
CA _H	ADC_RESR0L Reset: 00_H Result Register 0 Low	Bit Field	RESULT			VF	DRC	0	CHNR	
		Type	rh			rh	rh	r	rh	
CB _H	ADC_RESR0H Reset: 00_H Result Register 0 High	Bit Field	RESULT							
		Type	rh							
CC _H	ADC_RESR1L Reset: 00_H Result Register 1 Low	Bit Field	RESULT			VF	DRC	0	CHNR	
		Type	rh			rh	rh	r	rh	
CD _H	ADC_RESR1H Reset: 00_H Result Register 1 High	Bit Field	RESULT							
		Type	rh							
CE _H	ADC_RESR2L Reset: 00_H Result Register 2 Low	Bit Field	RESULT			VF	DRC	0	CHNR	
		Type	rh			rh	rh	r	rh	
CF _H	ADC_RESR2H Reset: 00_H Result Register 2 High	Bit Field	RESULT							
		Type	rh							
D2 _H	ADC_RESR3L Reset: 00_H Result Register 3 Low	Bit Field	RESULT			VF	DRC	0	CHNR	
		Type	rh			rh	rh	r	rh	
D3 _H	ADC_RESR3H Reset: 00_H Result Register 3 High	Bit Field	RESULT							
		Type	rh							
RMAP = 0, PAGE 4										
CA _H	ADC_RCR0 Reset: 00_H Result Control Register 0	Bit Field	VFCT R	WFR	0	IEN	0	DLPF	0	DRCT R
		Type	rw	rw	r	rw	r	rw	r	rw

Memory Organization
Table 3-5 ADC Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
CB _H	ADC_RCR1 Reset: 00_H Result Control Register 1	Bit Field	VFCT R	WFR	0	IEN	0	DLPF	0	DRCT R
		Type	rw	rw	r	rw	r	rw	r	rw
CC _H	ADC_RCR2 Reset: 00_H Result Control Register 2	Bit Field	VFCT R	WFR	0	IEN	0	DLPF	0	DRCT R
		Type	rw	rw	r	rw	r	rw	r	rw
CD _H	ADC_RCR3 Reset: 00_H Result Control Register 3	Bit Field	VFCT R	WFR	0	IEN	0	DLPF	0	DRCT R
		Type	rw	rw	r	rw	r	rw	r	rw
CE _H	ADC_VFCR Reset: 00_H Valid Flag Clear Register	Bit Field	0				VFC3	VFC2	VFC1	VFC0
		Type	r				w	w	w	w
CF _H	ADC_ALR0 Reset: 00_H Alias Register 0	Bit Field	0					0	ALIAS0	
		Type	r					r	rw	
D2 _H	ADC_CNF Reset: 00_H Configure Out of Range Comparator Register	Bit Field	0				CNF3	CNF2	CNF1	CNF0
		Type	r				rw	rw	rw	rw
D3 _H	ADC_ETRCR Reset: 00_H External Trigger Control Register	Bit Field	0	ETRSEL1			ETRSEL0			
		Type	r	rw			rw			
RMAP = 0, PAGE 5										
CA _H	ADC_CHINFR Reset: 00_H Channel Interrupt Flag Register	Bit Field	0				CHINF 3	CHINF 2	CHINF 1	CHINF 0
		Type	r				rh	rh	rh	rh
CB _H	ADC_CHINCR Reset: 00_H Channel Interrupt Clear Register	Bit Field	0				CHINC 3	CHINC 2	CHINC 1	CHINC 0
		Type	r				w	w	w	w
CC _H	ADC_CHINSR Reset: 00_H Channel Interrupt Set Register	Bit Field	0				CHINS 3	CHINS 2	CHINS 1	CHINS 0
		Type	r				w	w	w	w
CE _H	ADC_EVINFR Reset: 00_H Event Interrupt Flag Register	Bit Field	EVINF 7	EVINF 6	EVINF 5	EVINF 4	0		EVINF 1	EVINF 0
		Type	rh	rh	rh	rh	r		rh	rh
CF _H	ADC_EVINCR Reset: 00_H Event Interrupt Clear Flag Register	Bit Field	EVINC 7	EVINC 6	EVINC 5	EVINC 4	0		EVINC 1	EVINC 0
		Type	w	w	w	w	r		w	w
D2 _H	ADC_EVINSR Reset: 00_H Event Interrupt Set Flag Register	Bit Field	EVINS 7	EVINS 6	EVINS 5	EVINS 4	0		EVINS 1	EVINS 0
		Type	w	w	w	w	r		w	w
RMAP = 0, PAGE 6										
CA _H	ADC_CRCR1 Reset: 00_H Conversion Request Control Register 1	Bit Field	0				CH3	CH2	CH1	CH0
		Type	r				rwh	rwh	rwh	rwh
CB _H	ADC_CRPR1 Reset: 00_H Conversion Request Pending Register 1	Bit Field	0				CHP3	CHP2	CHP1	CHP0
		Type	r				rwh	rwh	rwh	rwh

Memory Organization
Table 3-5 ADC Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
CC _H	ADC_CRMR1 Reset: 00 _H Conversion Request Mode Register 1	Bit Field	0	LDEV	CLRP ND	SCAN	ENSI	ENTR	0	ENGT
		Type	r	w	w	rw	rw	rw	r	rw
CD _H	ADC_QMR0 Reset: 00 _H Queue Mode Register 0	Bit Field	CEV	TREV	FLUS H	CLRV	0	ENTR	0	ENGT
		Type	w	w	w	w	r	rw	r	rw
CE _H	ADC_QSR0 Reset: 20 _H Queue Status Register 0	Bit Field	0		EMPT Y	EV	0		FILL	
		Type	r		rh	rh	r		rh	
CF _H	ADC_Q0R0 Reset: 00 _H Queue 0 Register 0	Bit Field	EXTR	ENSI	RF	V	0		REQCHNR	
		Type	rh	rh	rh	rh	r		rh	
D2 _H	ADC_QBUR0 Reset: 00 _H Queue Backup Register 0	Bit Field	EXTR	ENSI	RF	V	0		REQCHNR	
		Type	rh	rh	rh	rh	r		rh	
D2 _H	ADC_QINR0 Reset: 00 _H Queue Input Register 0	Bit Field	EXTR	ENSI	RF	0			REQCHNR	
		Type	w	w	w	r			rh	

3.4.5.6 LEDSCU Registers

The LEDSCU SFRs can be accessed in the standard memory area (RMAP = 0).

Table 3-6 LEDSCU Register Overview

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
RMAP = 0										
97 _H	LTS_GLOBCTL0 Reset: 00 _H Global Control Register 0	Bit Field	LD_EN	TS_EN	CLK_PS					
		Type	rw	rw	rw					
D4 _H	LTS_COMPARE Reset: 00 _H Time Slice Compare Shadow Register	Bit Field	SHD_CMP							
		Type	rw							
D5 _H	LTS_LDLINE Reset: 00 _H LED Line Pattern Shadow Register	Bit Field	SHD_LINE							
		Type	rw							
D6 _H	LTS_LDTSCTL Reset: 00 _H LED and Touch-sense Control Register	Bit Field	NR_LEDCOL			COL LEV	NR_PADT			TSO EXT
		Type	rw			rw	rw			rw
D7 _H	LTS_TSCTL Reset: 00 _H Touch-sense Control Register	Bit Field	TS CTR OVL	TS CTRR	TS CTR SAT	E PULL	PADT SW	PADT		
		Type	rw	rw	rw	rw	rw	rwh		
D8 _H	LTS_GLOBCTL1 Reset: 00 _H Global Control Register 1	Bit Field	TSF	ITS_EN	TFF	ITF_EN	CLK SEL	FNCOL		
		Type	rwh	rw	rwh	rw	rw	rh		
D9 _H	LTS_TSVAL Reset: 00 _H Touch-sense Counter Value Register	Bit Field	TSCTRVAL							
		Type	rwh							

3.4.5.7 RTC Registers

The RTC SFRs can be accessed in the standard memory area (RMAP = 0).

Table 3-7 RTC Register Overview

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
RMAP = 0										
95 _H	RTC_RTCON Reset: 00_H Real-Time Clock Control Register	Bit Field	SFRT C	CRFT C	ESRT C	ECRT C	RTCC T	RTM		RTCC
		Type	rwh	rwh	rw	rw	rwh	rw		rw
96 _H	RTC_RTCON1 Reset: 02_H Real-Time Clock Control Register 1	Bit Field	0							RTYR
		Type	r							rw
E1 _H	RTC_CNT0 Reset: 00_H Clock Count Register 0 Mode 0	Bit Field	0		MILLISECONDS					
		Type	r		rwh					
E1 _H	RTC_CNT0 Reset: 00_H Clock Count Register 0 Modes 1 and 3	Bit Field	CNT_VAL							
		Type	rwh							
E2 _H	RTC_CNT1 Reset: 00_H Clock Count Register 1 Modes 0 and 2	Bit Field	0		SECONDS					
		Type	r		rwh					
E2 _H	RTC_CNT1 Reset: 00_H Clock Count Register 1 Modes 1 and 3	Bit Field	CNT_VAL							
		Type	rwh							
E3 _H	RTC_CNT2 Reset: 00_H Clock Count Register 2 Modes 0 and 2	Bit Field	0		MINUTES					
		Type	r		rwh					
E3 _H	RTC_CNT2 Reset: 00_H Clock Count Register 2 Modes 1 and 3	Bit Field	CNT_VAL							
		Type	rwh							
E4 _H	RTC_CNT3 Reset: 00_H Clock Count Register 3 Modes 0 and 2	Bit Field	0			HOURS				
		Type	r			rwh				
E4 _H	RTC_CNT3 Reset: 00_H Clock Count Register 3 Modes 1 and 3	Bit Field	CNT_VAL							
		Type	rwh							
E5 _H	RTC_CNT4 Reset: 00_H Clock Count Register 4 Modes 0 and 2	Bit Field	DAYS							
		Type	rwh							
E6 _H	RTC_CNT5 Reset: 00_H Clock Count Register 5 Modes 0 and 2	Bit Field	0							DAYS
		Type	r							rw
E7 _H	RTC_RTCCR0 Reset: 00_H Real-Time Clock Compare/Capture Register 0 Mode 0	Bit Field	0		CC_MSECS					
		Type	r		rwh					
E7 _H	RTC_RTCCR0 Reset: 00_H Real-Time Clock Compare/Capture Register 0 Modes 1 and 3	Bit Field	CC_VAL							
		Type	rwh							

Memory Organization
Table 3-7 RTC Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
E9 _H	RTC_RTCCR1 Reset: 00_H Real-Time Clock Compare/Capture Register 1 Modes 0 and 2	Bit Field	0		CC_SECONDS					
		Type	r		rwh					
E9 _H	RTC_RTCCR1 Reset: 00_H Real-Time Clock Compare/Capture Register 1 Modes 1 and 3	Bit Field	CC_VAL							
		Type	rwh							
EA _H	RTC_RTCCR2 Reset: 00_H Real-Time Clock Compare/Capture Register 2 Modes 0 and 2	Bit Field	0		CC_MINUTES					
		Type	r		rwh					
EA _H	RTC_RTCCR2 Reset: 00_H Real-Time Clock Compare/Capture Register 2 Modes 1 and 3	Bit Field	CC_VAL							
		Type	rwh							
EB _H	RTC_RTCCR3 Reset: 00_H Real-Time Clock Compare/Capture Register 3 Modes 0 and 2	Bit Field	0		CC_HOURS					
		Type	r		rwh					
EB _H	RTC_RTCCR3 Reset: 00_H Real-Time Clock Compare/Capture Register 3 Modes 1 and 3	Bit Field	CC_VAL							
		Type	rwh							
EC _H	RTC_RTCCR4 Reset: 00_H Real-Time Clock Compare/Capture Register 4 Modes 0 and 2	Bit Field	CC_DAYS							
		Type	rwh							
ED _H	RTC_RTCCR5 Reset: 00_H Real-Time Clock Compare/Capture Register 5 Modes 0 and 2	Bit Field	0							CC_D AYS
		Type	r							rw

Memory Organization

3.4.5.8 Timer 2 Registers

The Timer 2 SFRs can be accessed in the standard memory area (RMAP = 0).

Table 3-8 T2 Register Overview

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
RMAP = 0										
C0 _H	T2_T2CON Reset: 00_H Timer 2 Control Register	Bit Field	TF2	EXF2	0		EXEN 2	TR2	C_T2	CP_RL 2
		Type	rwh	rwh	r		rw	rwh	rw	rw
C1 _H	T2_T2MOD Reset: 00_H Timer 2 Mode Register	Bit Field	T2RE GS	T2RH EN	EDGE SEL	PREN	T2PRE			DCEN
		Type	rw	rw	rw	rw	rw	rw	rw	rw
C2 _H	T2_RC2L Reset: 00_H Timer 2 Reload/Capture Register Low	Bit Field	RC2							
		Type	rwh							
C3 _H	T2_RC2H Reset: 00_H Timer 2 Reload/Capture Register High	Bit Field	RC2							
		Type	rwh							
C4 _H	T2_T2L Reset: 00_H Timer 2 Register Low	Bit Field	THL2							
		Type	rwh							
C5 _H	T2_T2H Reset: 00_H Timer 2 Register High	Bit Field	THL2							
		Type	rwh							
C6 _H	T2_T2CON1 Reset: 03_H Timer 2 Control Register 1	Bit Field	0						TF2EN	EXF2E N
		Type	r						rw	rw

3.4.5.9 CCU6 Registers

The CCU6 SFRs can be accessed in the standard memory area (RMAP = 0).

Table 3-9 CCU6 Register Overview

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
RMAP = 0										
A3 _H	CCU6_PAGE Reset: 00_H Page Register	Bit Field	OP		STNR		0	PAGE		
		Type	w		w		r	rw		
RMAP = 0, PAGE 0										
9A _H	CCU6_CC63SRL Reset: 00_H Capture/Compare Shadow Register for Channel CC63 Low	Bit Field	CC63SL							
		Type	rw							
9B _H	CCU6_CC63SRH Reset: 00_H Capture/Compare Shadow Register for Channel CC63 High	Bit Field	CC63SH							
		Type	rw							
9C _H	CCU6_TCTR4L Reset: 00_H Timer Control Register 4 Low	Bit Field	T12 STD	T12 STR	0		DT RES	T12 RES	T12R S	T12R R
		Type	w	w	r		w	w	w	w

Memory Organization
Table 3-9 CCU6 Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
9D _H	CCU6_TCTR4H Reset: 00 _H Timer Control Register 4 High	Bit Field	T13 STD	T13 STR	0			T13 RES	T13R S	T13R R
		Type	w	w	r			w	w	w
9E _H	CCU6_MCMOUTSL Reset: 00 _H Multi-Channel Mode Output Shadow Register Low	Bit Field	STRM CM	0	MCMPS					
		Type	w	r	rw					
9F _H	CCU6_MCMOUTSH Reset: 00 _H Multi-Channel Mode Output Shadow Register High	Bit Field	STRH P	0	CURHS			EXPHS		
		Type	w	r	rw			rw		
A4 _H	CCU6_ISRL Reset: 00 _H Capture/Compare Interrupt Status Reset Register Low	Bit Field	RT12 PM	RT12 OM	RCC6 2F	RCC6 2R	RCC6 1F	RCC6 1R	RCC6 0F	RCC6 0R
		Type	w	w	w	w	w	w	w	w
A5 _H	CCU6_ISRH Reset: 00 _H Capture/Compare Interrupt Status Reset Register High	Bit Field	RSTR	RIDLE	RWH E	RCHE	0	RTRP F	RT13 PM	RT13 CM
		Type	w	w	w	w	r	w	w	w
A6 _H	CCU6_CMPMODIFL Reset: 00 _H Compare State Modification Register Low	Bit Field	0	MCC6 3S	0			MCC6 2S	MCC6 1S	MCC6 0S
		Type	r	w	r			w	w	w
A7 _H	CCU6_CMPMODIFH Reset: 00 _H Compare State Modification Register High	Bit Field	0	MCC6 3R	0			MCC6 2R	MCC6 1R	MCC6 0R
		Type	r	w	r			w	w	w
FA _H	CCU6_CC60SRL Reset: 00 _H Capture/Compare Shadow Register for Channel CC60 Low	Bit Field	CC60SL							
		Type	rwh							
FB _H	CCU6_CC60SRH Reset: 00 _H Capture/Compare Shadow Register for Channel CC60 High	Bit Field	CC60SH							
		Type	rwh							
FC _H	CCU6_CC61SRL Reset: 00 _H Capture/Compare Shadow Register for Channel CC61 Low	Bit Field	CC61SL							
		Type	rwh							
FD _H	CCU6_CC61SRH Reset: 00 _H Capture/Compare Shadow Register for Channel CC61 High	Bit Field	CC61SH							
		Type	rwh							
FE _H	CCU6_CC62SRL Reset: 00 _H Capture/Compare Shadow Register for Channel CC62 Low	Bit Field	CC62SL							
		Type	rwh							
FF _H	CCU6_CC62SRH Reset: 00 _H Capture/Compare Shadow Register for Channel CC62 High	Bit Field	CC62SH							
		Type	rwh							
RMAP = 0, PAGE 1										
9A _H	CCU6_CC63RL Reset: 00 _H Capture/Compare Register for Channel CC63 Low	Bit Field	CC63VL							
		Type	rh							
9B _H	CCU6_CC63RH Reset: 00 _H Capture/Compare Register for Channel CC63 High	Bit Field	CC63VH							
		Type	rh							
9C _H	CCU6_T12PRL Reset: 00 _H Timer T12 Period Register Low	Bit Field	T12PVL							
		Type	rwh							

Memory Organization
Table 3-9 CCU6 Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
9D _H	CCU6_T12PRH Reset: 00_H Timer T12 Period Register High	Bit Field	T12PVH							
		Type	rwh							
9E _H	CCU6_T13PRL Reset: 00_H Timer T13 Period Register Low	Bit Field	T13PVL							
		Type	rwh							
9F _H	CCU6_T13PRH Reset: 00_H Timer T13 Period Register High	Bit Field	T13PVH							
		Type	rwh							
A4 _H	CCU6_T12DTCL Reset: 00_H Dead-Time Control Register for Timer T12 Low	Bit Field	DTM							
		Type	rw							
A5 _H	CCU6_T12DTC Reset: 00_H Dead-Time Control Register for Timer T12 High	Bit Field	0	DTR2	DTR1	DTR0	0	DTE2	DTE1	DTE0
		Type	r	rh	rh	rh	r	rw	rw	rw
A6 _H	CCU6_TCTR0L Reset: 00_H Timer Control Register 0 Low	Bit Field	CTM	CDIR	STE1 2	T12R	T12 PRE	T12CLK		
		Type	rw	rh	rh	rh	rw	rw		
A7 _H	CCU6_TCTR0H Reset: 00_H Timer Control Register 0 High	Bit Field	0		STE1 3	T13R	T13 PRE	T13CLK		
		Type	r		rh	rh	rw	rw		
FA _H	CCU6_CC60RL Reset: 00_H Capture/Compare Register for Channel CC60 Low	Bit Field	CC60VL							
		Type	rh							
FB _H	CCU6_CC60RH Reset: 00_H Capture/Compare Register for Channel CC60 High	Bit Field	CC60VH							
		Type	rh							
FC _H	CCU6_CC61RL Reset: 00_H Capture/Compare Register for Channel CC61 Low	Bit Field	CC61VL							
		Type	rh							
FD _H	CCU6_CC61RH Reset: 00_H Capture/Compare Register for Channel CC61 High	Bit Field	CC61VH							
		Type	rh							
FE _H	CCU6_CC62RL Reset: 00_H Capture/Compare Register for Channel CC62 Low	Bit Field	CC62VL							
		Type	rh							
FF _H	CCU6_CC62RH Reset: 00_H Capture/Compare Register for Channel CC62 High	Bit Field	CC62VH							
		Type	rh							
RMAP = 0, PAGE 2										
9A _H	CCU6_T12MSELL Reset: 00_H T12 Capture/Compare Mode Select Register Low	Bit Field	MSEL61				MSEL60			
		Type	rw				rw			
9B _H	CCU6_T12MSELH Reset: 00_H T12 Capture/Compare Mode Select Register High	Bit Field	DBYP	HSYNC			MSEL62			
		Type	rw	rw			rw			
9C _H	CCU6_IENL Reset: 00_H Capture/Compare Interrupt Enable Register Low	Bit Field	ENT1 2 PM	ENT1 2 OM	ENCC 62F	ENCC 62R	ENCC 61F	ENCC 61R	ENCC 60F	ENCC 60R
		Type	rw	rw	rw	rw	rw	rw	rw	rw

Memory Organization
Table 3-9 CCU6 Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
9D _H	CCU6_IENH Reset: 00_H Capture/Compare Interrupt Enable Register High	Bit Field	EN STR	EN IDLE	EN WHE	EN CHE	0	EN TRPF	ENT1 3PM	ENT1 3CM
		Type	rw	rw	rw	rw	r	rw	rw	rw
9E _H	CCU6_INPL Reset: 40_H Capture/Compare Interrupt Node Pointer Register Low	Bit Field	INPCHE		INPCC62		INPCC61		INPCC60	
		Type	rw		rw		rw		rw	
9F _H	CCU6_INPH Reset: 39_H Capture/Compare Interrupt Node Pointer Register High	Bit Field	0		INPT13		INPT12		INPERR	
		Type	r		rw		rw		rw	
A4 _H	CCU6_ISSL Reset: 00_H Capture/Compare Interrupt Status Set Register Low	Bit Field	ST12 PM	ST12 OM	SCC6 2F	SCC6 2R	SCC6 1F	SCC6 1R	SCC6 0F	SCC6 0R
		Type	w	w	w	w	w	w	w	w
A5 _H	CCU6_ISSH Reset: 00_H Capture/Compare Interrupt Status Set Register High	Bit Field	SSTR	SIDLE	SWHE	SCHE	SWH C	STRP F	ST13 PM	ST13 CM
		Type	w	w	w	w	w	w	w	w
A6 _H	CCU6_PSLR Reset: 00_H Passive State Level Register	Bit Field	PSL63	0	PSL					
		Type	rwh	r	rwh					
A7 _H	CCU6_MCMCTR Reset: 00_H Multi-Channel Mode Control Register	Bit Field	0		SWSYN		0	SWSEL		
		Type	r		rw		r	rw		
FA _H	CCU6_TCTR2L Reset: 00_H Timer Control Register 2 Low	Bit Field	0	T13TED		T13TEC			T13 SSC	T12 SSC
		Type	r	rw		rw			rw	rw
FB _H	CCU6_TCTR2H Reset: 00_H Timer Control Register 2 High	Bit Field	0				T13RSEL		T12RSEL	
		Type	r				rw		rw	
FC _H	CCU6_MODCTRL Reset: 00_H Modulation Control Register Low	Bit Field	MCM EN	0	T12MODEN					
		Type	rw	r	rw					
FD _H	CCU6_MODCTRH Reset: 00_H Modulation Control Register High	Bit Field	ECT1 3O	0	T13MODEN					
		Type	rw	r	rw					
FE _H	CCU6_TRPCTRL Reset: 00_H Trap Control Register Low	Bit Field	0					TRPM 2	TRPM 1	TRPM 0
		Type	r					rw	rw	rw
FF _H	CCU6_TRPCTRH Reset: 00_H Trap Control Register High	Bit Field	TRPP EN	TRPE N13	TRPEN					
		Type	rw	rw	rw					
RMAP = 0, PAGE 3										
9A _H	CCU6_MCMOUTL Reset: 00_H Multi-Channel Mode Output Register Low	Bit Field	0	R	MCMP					
		Type	r	rh	rh					
9B _H	CCU6_MCMOUTH Reset: 00_H Multi-Channel Mode Output Register High	Bit Field	0		CURH			EXPH		
		Type	r		rh			rh		

Memory Organization
Table 3-9 CCU6 Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
9C _H	CCU6_ISL Reset: 00_H Capture/Compare Interrupt Status Register Low	Bit Field	T12 PM	T12 OM	ICC62 F	ICC62 R	ICC61 F	ICC61 R	ICC60 F	ICC60 R
		Type	rh	rh	rh	rh	rh	rh	rh	rh
9D _H	CCU6_ISH Reset: 00_H Capture/Compare Interrupt Status Register High	Bit Field	STR	IDLE	WHE	CHE	TRPS	TRPF	T13 PM	T13 CM
		Type	rh	rh	rh	rh	rh	rh	rh	rh
9E _H	CCU6_PISEL0L Reset: 00_H Port Input Select Register 0 Low	Bit Field	ISTRP		ISCC62		ISCC61		ISCC60	
		Type	rw		rw		rw		rw	
9F _H	CCU6_PISEL0H Reset: 00_H Port Input Select Register 0 High	Bit Field	IST12HR		ISPOS2		ISPOS1		ISPOS0	
		Type	rw		rw		rw		rw	
A4 _H	CCU6_PISEL2 Reset: 00_H Port Input Select Register 2	Bit Field	0						IST13HR	
		Type	r						rw	
FA _H	CCU6_T12L Reset: 00_H Timer T12 Counter Register Low	Bit Field	T12CVL							
		Type	rwh							
FB _H	CCU6_T12H Reset: 00_H Timer T12 Counter Register High	Bit Field	T12CVH							
		Type	rwh							
FC _H	CCU6_T13L Reset: 00_H Timer T13 Counter Register Low	Bit Field	T13CVL							
		Type	rwh							
FD _H	CCU6_T13H Reset: 00_H Timer T13 Counter Register High	Bit Field	T13CVH							
		Type	rwh							
FE _H	CCU6_CMPSTATL Reset: 00_H Compare State Register Low	Bit Field	0	CC63 ST	CC POS2	CC POS1	CC POS0	CC62 ST	CC61 ST	CC60 ST
		Type	r	rh	rh	rh	rh	rh	rh	rh
FF _H	CCU6_CMPSTATH Reset: 00_H Compare State Register High	Bit Field	T13IM	COU63PS	COU62PS	CC62 PS	COU61PS	CC61 PS	COU60PS	CC60 PS
		Type	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Memory Organization

3.4.5.10 SSC Registers

The SSC SFRs can be accessed in the standard memory area (RMAP = 0).

Table 3-10 SSC Register Overview

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
RMAP = 0										
AA _H	SSC_CONL Reset: 00_H Control Register Low Programming Mode	Bit Field	LB	PO	PH	HB	BM			
		Type	rw	rw	rw	rw	rw			
AA _H	SSC_CONL Reset: 00_H Control Register Low Operating Mode	Bit Field	0				BC			
		Type	r				rh			
AB _H	SSC_CONH Reset: 00_H Control Register High Programming Mode	Bit Field	EN	MS	0	AREN	BEN	PEN	REN	TEN
		Type	rw	rw	r	rw	rw	rw	rw	rw
AB _H	SSC_CONH Reset: 00_H Control Register High Operating Mode	Bit Field	EN	MS	0	BSY	BE	PE	RE	TE
		Type	rw	rw	r	rh	rw	rw	rw	rw
AC _H	SSC_TBL Reset: 00_H Transmitter Buffer Register Low	Bit Field	TB_VALUE							
		Type	rw							
AD _H	SSC_RBL Reset: 00_H Receiver Buffer Register Low	Bit Field	RB_VALUE							
		Type	rh							
AE _H	SSC_BRL Reset: 00_H Baud Rate Timer Reload Register Low	Bit Field	BR_VALUE							
		Type	rw							
AF _H	SSC_BRH Reset: 00_H Baud Rate Timer Reload Register High	Bit Field	BR_VALUE							
		Type	rw							

3.4.5.11 IIC Registers

The IIC SFRs can be accessed in the standard memory area (RMAP = 0).

Table 3-11 IIC Register Overview

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
RMAP = 0										
DA _H	IIC_ADDR Reset: 00_H Slave Address Register	Bit Field	SLA							GCE
		Type	rw							rw
DB _H	IIC_DATA Reset: 00_H Data Byte Register	Bit Field	DATA							
		Type	rw							
DC _H	IIC_CNTR Reset: 00_H Control Register	Bit Field	IEN	ENAB	STA	STP	IFLG	AAK	0	
		Type	rw	rw	rwh	rwh	rwh	rw	r	
DD _H	IIC_STAT Reset: F8_H Status Register	Bit Field	STAT					0		
		Type	rh					r		

Memory Organization

Table 3-11 IIC Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
DD _H	IIC_BRCR Reset: 00_H Baud Rate Control Register	Bit Field	0	BRP				PREDIV		
		Type	w	w				w		
DE _H	IIC_ADDRX Reset: 00_H Extended Slave Address Register	Bit Field	SLAX							
		Type	rw							
DF _H	IIC_SRST Reset: UU_H Software Reset Register	Bit Field	SRST							
		Type	w							

3.4.5.12 OCDS Registers

The OCDS SFRs can be accessed in the mapped memory area (RMAP = 1).

Table 3-12 OCDS Register Overview

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
RMAP = 1										
E9 _H	MMCR2 Reset: 1U_H Monitor Mode Control 2 Register	Bit Field	STMO DE	0	DSUS P	MBCO N	ALTDI	MMEP	MMOD E	JENA
		Type	rw	rw	rw	rwh	rw	rwh	rh	rh
F1 _H	MMCR Reset: 00_H Monitor Mode Control Register	Bit Field	MEXIT _P	MEXIT	MMNM IE	MSTE P	MRAM S_P	MRAM S	TRF	RRF
		Type	w	rwh	rw	rw	w	rwh	rh	rh
F2 _H	MMSR Reset: 00_H Monitor Mode Status Register	Bit Field	0		EXBF	SWBF	HWB3 F	HWB2 F	HWB1 F	HWB0 F
		Type	rw		rwh	rwh	rwh	rwh	rwh	rwh
F3 _H	MMBPCR Reset: 00_H Breakpoints Control Register	Bit Field	SWBC	HWB3C		HWB2C		HWB1 C	HWB0C	
		Type	rw	rw		rw		rw	rw	
F4 _H	MMICR Reset: 00_H Monitor Mode Interrupt Control Register	Bit Field	DVEC T	DRET R	COMR ST	MSTS EL	FMIR W	FNMIR R	NMIR WE	NMIR RE
		Type	rwh	rwh	rwh	rh	rwh	rwh	rw	rw
F5 _H	MMDR Reset: 00_H Monitor Mode Data Transfer Register Receive	Bit Field	MMRR							
		Type	rh							
F5 _H	MMDR Reset: 00_H Monitor Mode Data Transfer Register Transmit	Bit Field	MMTR							
		Type	w							
F6 _H	HWBPSR Reset: 00_H Hardware Breakpoints Select Register	Bit Field	0			BPSEL _P	BPSEL			
		Type	r			w	rw			
F7 _H	HWBPDR Reset: 00_H Hardware Breakpoints Data Register	Bit Field	HWBPxx							
		Type	rw							

Memory Organization

Table 3-12 OCDS Register Overview (cont'd)

Addr	Register Name	Bit	7	6	5	4	3	2	1	0
EB _H	MMWR1 Reset: 00_H Monitor Work Register 1	Bit Field	MMWR1							
		Type	rw							
EC _H	MMWR2 Reset: 00_H Monitor Work Register 2	Bit Field	MMWR2							
		Type	rw							

4 Flash Memory

The XC82x has an embedded user-programmable non-volatile Flash memory that allows for fast and reliable storage of user code and data. It is operated with a single 2.5 V supply from the Embedded Voltage Regulator (EVR) and does not require additional programming or erasing voltage. The sectorization of the Flash memory allows each sector to be erased independently.

Features

- In-System Programming (ISP) via UART
- In-Application Programming (IAP)
- Error Correction Code (ECC) for dynamic correction of single-bit errors
- Background program and erase operations for CPU load minimization
- Support for aborting erase operation
- 32-byte minimum program width
- 1-sector minimum erase width
- 1-byte read access
- 1 or $3 \times$ CCLK period read access time (depending on zero or one wait state)
- Flash is delivered in erased state (read all zeros)

4.1 Flash Memory Address Mapping

The program memory map for the two Flash sizes is shown in [Figure 4-1](#).

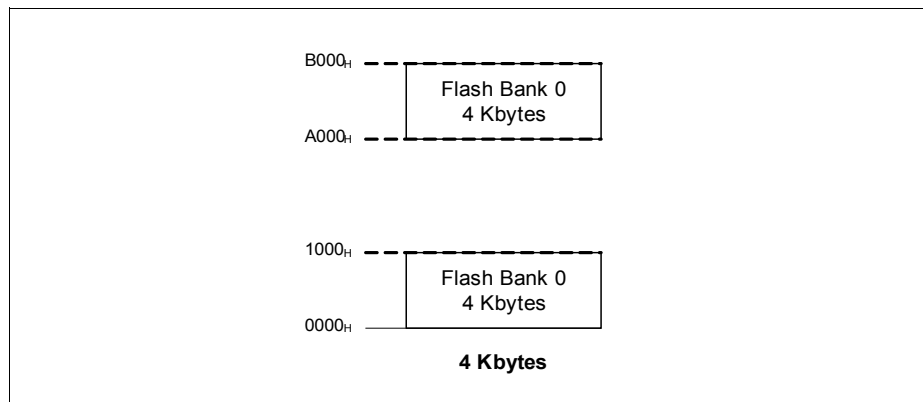


Figure 4-1 Flash Memory Map

For the XC82x, only a single 4-Kbyte Flash bank, Bank 0, is available. The Flash bank is mapped to two program memory address spaces 0000_H - 0FFF_H and A000_H - AFFF_H. The double-mapping of the Flash banks is intended to facilitate software coding. As a general guideline, the lower address spaces (0000_H - 0FFF_H) should be used for Flash contents that are intended to be used as program code. Whereas the higher address space (A000_H - AFFF_H) should be used for Flash contents that are intended to be used as data.

4.2 Flash Bank Sectorization

The XC82x Flash devices consist of one 4-Kbyte Flash bank with the sectorization shown in [Figure 4-2](#). Each Flash bank can be used for code and data storage.

Flash Memory

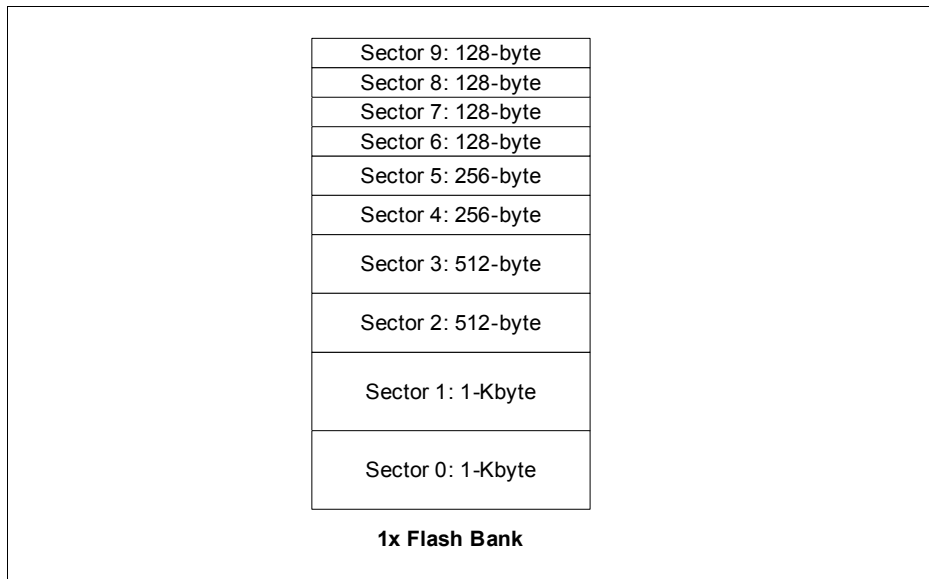


Figure 4-2 Flash Bank Sectorization

Sector Partitioning in each 4-Kbyte Flash bank:

- Two 1-Kbyte sectors
- Two 512-byte sectors
- Two 256-byte sectors
- Four 128-byte sectors

The internal structure of each Flash bank represents a sector architecture for flexible erase capability. The minimum erase width is always a complete sector, and sectors can be erased separately or in parallel. Contrary to standard EEPROMs, erased Flash memory cells contain 0s.

The Flash bank is divided into more physical sectors for extended erasing and reprogramming capability; even numbers for each sector size are provided to allow greater flexibility and the ability to adapt to a wide range of application requirements.

For example, the user's program can implement a buffer mechanism for each sector. Double copies of each data set can be stored in separate sectors of similar size to ensure that a backup copy of the data set is available in the event the actual data set is corrupted or erased.

Alternatively, the user can implement an algorithm for EEPROM emulation, which uses the Flash bank like a circular stack memory; the latest data updates are always programmed on top of the actual region. When the top of the sector is reached, all actual

Flash Memory

data (representing the EEPROM data) is copied to the bottom area of the next sector and the last sector is then erased. This round robin procedure, using multifold replications of the emulated EEPROM size, significantly increases the Flash endurance. To speed up data search, the RAM can be used to contain the pointer to the valid data set.

4.3 Wordline Address

The wordline (WL) addresses of Flash Bank 0 are given in **Figure 4-3**.

Flash Bank 0				Flash Bank 0 (Alternate Address Mapping)			
Byte 31	Byte 2	Byte 1	Byte 0	Byte 31	Byte 2	Byte 1	Byte 0
0FF _{FH}	0FE _{2H}	0FE _{1H}	0FE _{0H}	AFF _{FH}	AFE _{2H}	AFE _{1H}	AFE _{0H}
...
0F9 _{FH}	0F8 _{2H}	0F8 _{1H}	0F8 _{0H}	AF9 _{FH}	AF8 _{2H}	AF8 _{1H}	AF8 _{0H}
0F7 _{FH}	0F6 _{2H}	0F6 _{1H}	0F6 _{0H}	AF7 _{FH}	AF6 _{2H}	AF6 _{1H}	AF6 _{0H}
...
0F1 _{FH}	0F0 _{2H}	0F0 _{1H}	0F0 _{0H}	AF1 _{FH}	AF0 _{2H}	AF0 _{1H}	AF0 _{0H}
0EF _{FH}	0EE _{2H}	0EE _{1H}	0EE _{0H}	AFF _{FH}	AEE _{2H}	AEE _{1H}	AEE _{0H}
...
0E9 _{FH}	0E8 _{2H}	0E8 _{1H}	0E8 _{0H}	AE9 _{FH}	AE8 _{2H}	AE8 _{1H}	AE8 _{0H}
0E7 _{FH}	0E6 _{2H}	0E6 _{1H}	0E6 _{0H}	AE7 _{FH}	AE6 _{2H}	AE6 _{1H}	AE6 _{0H}
...
0E1 _{FH}	0E0 _{2H}	0E0 _{1H}	0E0 _{0H}	AE1 _{FH}	AE0 _{2H}	AE0 _{1H}	AE0 _{0H}
0DF _{FH}	0DE _{2H}	0DE _{1H}	0DE _{0H}	ADF _{FH}	ADE _{2H}	ADE _{1H}	ADE _{0H}
...
0D1 _{FH}	0D0 _{2H}	0D0 _{1H}	0D0 _{0H}	AD1 _{FH}	AD0 _{2H}	AD0 _{1H}	AD0 _{0H}
0CF _{FH}	0CE _{2H}	0CE _{1H}	0CE _{0H}	ACF _{FH}	ACE _{2H}	ACE _{1H}	ACE _{0H}
...
0C1 _{FH}	0C0 _{2H}	0C0 _{1H}	0C0 _{0H}	AC1 _{FH}	AC0 _{2H}	AC0 _{1H}	AC0 _{0H}
0BF _{FH}	0BE _{2H}	0BE _{1H}	0BE _{0H}	ABF _{FH}	ABE _{2H}	ABE _{1H}	ABE _{0H}
...
0A3 _{FH}	0A2 _{2H}	0A2 _{1H}	0A2 _{0H}	AA3 _{FH}	AA2 _{2H}	AA2 _{1H}	AA2 _{0H}
0A1 _{FH}	0A0 _{2H}	0A0 _{1H}	0A0 _{0H}	AA1 _{FH}	AA0 _{2H}	AA0 _{1H}	AA0 _{0H}
09F _{FH}	09E _{2H}	09E _{1H}	09E _{0H}	A9F _{FH}	A9E _{2H}	A9E _{1H}	A9E _{0H}
...
083 _{FH}	082 _{2H}	082 _{1H}	082 _{0H}	A83 _{FH}	A82 _{2H}	A82 _{1H}	A82 _{0H}
081 _{FH}	080 _{2H}	080 _{1H}	080 _{0H}	A81 _{FH}	A80 _{2H}	A80 _{1H}	A80 _{0H}
07F _{FH}	07E _{2H}	07E _{1H}	07E _{0H}	A7F _{FH}	A7E _{2H}	A7E _{1H}	A7E _{0H}
...
045 _{FH}	044 _{2H}	044 _{1H}	044 _{0H}	A45 _{FH}	A44 _{2H}	A44 _{1H}	A44 _{0H}
043 _{FH}	042 _{2H}	042 _{1H}	042 _{0H}	A43 _{FH}	A42 _{2H}	A42 _{1H}	A42 _{0H}
041 _{FH}	040 _{2H}	040 _{1H}	040 _{0H}	A41 _{FH}	A40 _{2H}	A40 _{1H}	A40 _{0H}
03F _{FH}	03E _{2H}	03E _{1H}	03E _{0H}	A3F _{FH}	A3E _{2H}	A3E _{1H}	A3E _{0H}
...
005 _{FH}	004 _{2H}	004 _{1H}	004 _{0H}	A05 _{FH}	A04 _{2H}	A04 _{1H}	A04 _{0H}
003 _{FH}	002 _{2H}	002 _{1H}	002 _{0H}	A03 _{FH}	A02 _{2H}	A02 _{1H}	A02 _{0H}
001 _{FH}	000 _{2H}	000 _{1H}	000 _{0H}	A01 _{FH}	A00 _{2H}	A00 _{1H}	A00 _{0H}
WL Address				WL Address			

Figure 4-3 Flash Bank 0 Wordline Addresses

Flash Memory

A WL address can be calculated as follow:

$$0000_H/A000_H + 20_H \times n, \text{ with } 0 \leq n \leq 127 \text{ for Flash Bank 0} \quad (4.1)$$

Only one out of all the wordlines in the Flash banks can be programmed each time. The maximum/minimum program width of each WL is 32 bytes. Before programming can be done, the user must first write the number of bytes of data that is equivalent to the program width into the IRAM using 'MOV' instructions. Then, the Boot-loader (BSL) routine (see [Section 4.6](#)) or Flash program subroutine (see [Section 4.7.1](#)) will transfer this IRAM data to the corresponding write buffers of the targeted Flash bank. Once the data are assembled in the write buffers, the charge pump voltages are ramped up by a built-in program and erase state machine. Once the voltage ramping is completed, the volatile data content in the write buffers would have been stored into the non-volatile Flash cells along the selected WL. The WL is selected via the WL addresses shown in [Figure 4-3](#). It is necessary to fill the IRAM with the number of bytes of data as defined by the program width, otherwise the previous values stored in the write buffers will remain and be programmed into the WL.

The same WL can be programmed twice before erasing is required as the Flash cells are able to withstand two gate disturbs. This means if the number of data bytes that need to be written is smaller than the 32 bytes minimum programming width, the user can opt to program this number of data bytes (x; where x can be any integer from 1 to 31) first and program the remaining bytes (32-x) later. However, since the minimum programming width of Flash is always 32 bytes, the bytes that are unused in each programming cycle must be written with all zeros.

Figure 4-4 shows an example of programming the same wordline twice with 16 bytes of data. In the first program cycle, the lower 16 bytes are written with valid data while the upper 16 bytes that do not contain meaningful data are written with all zeros. In the second program cycle, it will be opposite as now only the upper 16 bytes can be written with valid data and the lower 16 bytes, which already contain meaningful data, must be written with all zeros.

Flash Memory

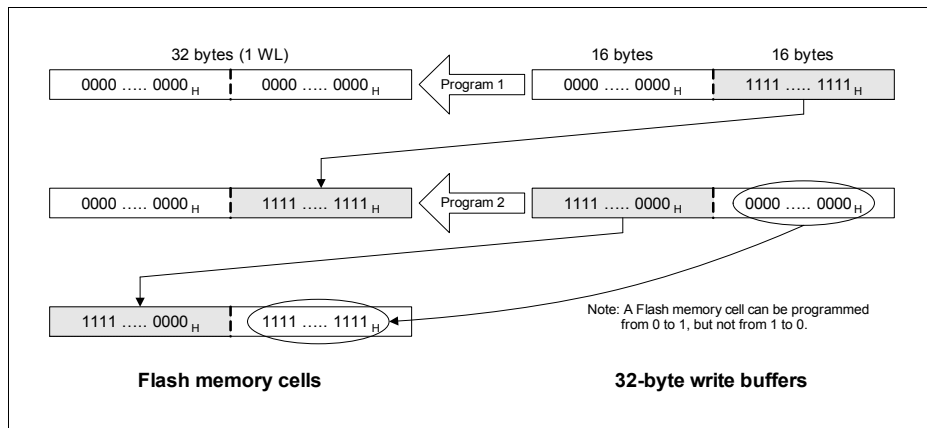


Figure 4-4 Flash Program

4.4 Operating Modes

The Flash operating modes for each bank are shown in [Figure 4-5](#).

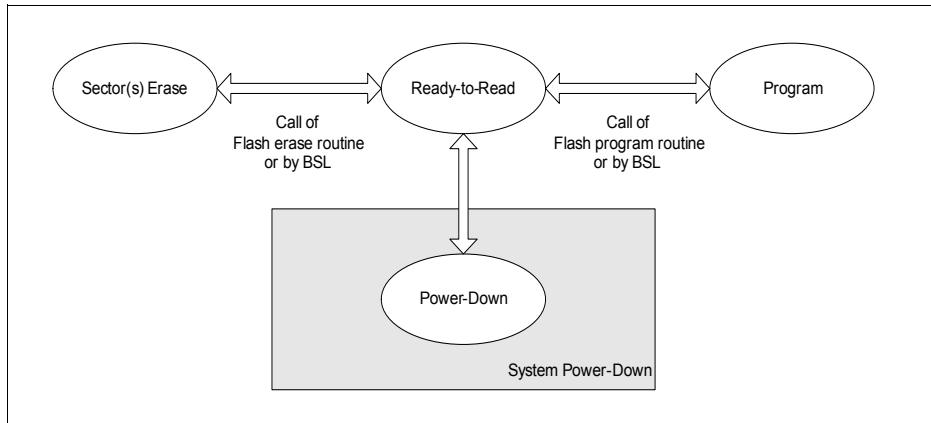


Figure 4-5 Flash Operating Modes

In general, the Flash operating modes are controlled by the BSL and Flash program/erase subroutines (see [Section 4.7](#)).

Each Flash bank must be in ready-to-read mode before the program mode or sector(s) erase mode is entered. In the ready-to-read mode, the 32-byte write buffers for each Flash bank can be written and the memory cell contents read via CPU access. In the program mode, data in the 32-byte write buffers is programmed into the Flash memory cells of the targeted wordline.

The operating modes for each Flash bank are enforced by its dedicated state machine to ensure the correct sequence of Flash mode transition. This avoids inadvertent destruction of the Flash contents with a reasonably low software overhead. The state machine also ensures that a Flash bank is blocked (no read access possible) while it is being programmed or erased. At any time, a Flash bank can only be in ready-to-read, program or sector(s) erase mode. However, it is possible to program/erase one Flash bank while reading from another.

When the user sets bit `PMCON0.PD = 1` to enter the system power-down mode, the Flash banks are automatically brought to its power-down state by hardware. Upon wake-up from system power-down, the Flash banks are brought to ready-to-read mode to allow access by the CPU.

4.5 Error Detection and Correction

The 8-bit data from the CPU is encoded with an Error Correction Code (ECC) before being stored in the Flash memory. During a read access, data is retrieved from the Flash memory and decoded for dynamic error detection and correction.

The correction algorithm (hamming code) has the capability to:

- Detect and correct all 1-bit errors
- Detect all 2-bit errors, but cannot correct

No distinction is made between a corrected 1-bit error (result is valid) and an uncorrected 2-bit error (result is invalid). In both cases, an ECC non-maskable interrupt (NMI) event is generated; bit FNMIECC in register NMISR is set, and if enabled via NMICON.NMIECC, an NMI to the CPU is triggered. The 16-bit Flash address at which the ECC error occurs is stored in the system control SFRs FEAL and FEAH, and can be accessed by the interrupt service routine to determine the Flash bank/sector in which the error occurred.

Flash Memory
4.5.1 Flash Error Address Register

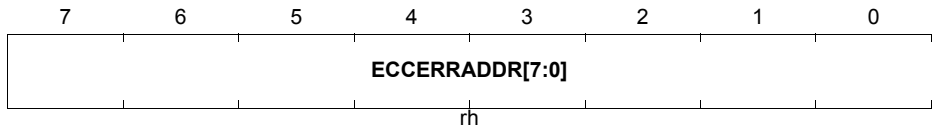
The FEAL and FEAH registers together store the 16-bit Flash address at which the ECC error occurs. The bit field PAGE of SCU_PAGE register must be programmed before accessing these registers.

FEAL

Flash Error Address Register, Low byte(F6_H)

Reset Value: 00_H

RMAP: 0, PAGE: 5



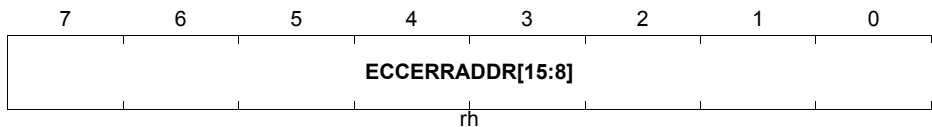
Field	Bits	Type	Description
ECCERRADDR	[7:0]	rh	ECC Error Address Value [7:0]

FEAH

Flash Error Address Register, High byte(F7_H)

Reset Value: 00_H

RMAP: 0, PAGE: 5



Field	Bits	Type	Description
ECCERRADDR	[7:0]	rh	ECC Error Address Value [15:8]

4.6 In-System Programming

In-System Programming (ISP) of the Flash memory is supported via the Boot ROM-based Boot-loader (BSL), allowing a blank microcontroller device mounted onto an application board to be programmed with the user code, and also a previously programmed device to be erased then reprogrammed without removal from the board. This feature offers ease-of-use and versatility for the embedded design.

ISP is supported through the microcontroller's serial interface (UART) which is connected to the personal computer host via the commonly available RS-232 serial cable. The BSL mode is selected if the programmed BMI and BMI values match the corresponding values defined for the BSL mode after power-on or hardware reset. The BSL routine will first perform an automatic synchronization with the transfer speed (baud rate) of the serial communication partner (personal computer host). Communication between the BSL routine and the host is done via a transfer protocol; information is sent from the host to the microcontroller in blocks with specified block structure, and the BSL routine acknowledges the received data by returning a single acknowledge or error byte. User can program, erase or execute the Flash bank(s).

The available working modes include:

- Transfer user program from host to Flash
- Execute user program in Flash
- Erase Flash sector(s)
- Mass Erase of all Flash sectors

4.7 In-Application Programming

In some applications, the Flash contents may need to be modified during program execution. In-Application Programming (IAP) is supported so that users can program or erase the Flash memory from their Flash user program by calling some subroutines in the Boot ROM (see [Figure 4-6](#) and [Chapter 22](#)). The Flash subroutines will first perform some checks and an initialization sequence before starting the program or erase operation. A manual check on the Flash data is necessary to determine if the programming or erasing was successful by using the 'MOVC' instruction to read out the Flash contents. Other special subroutines include aborting the Flash erase operation and checking the Flash bank ready-to-read status.

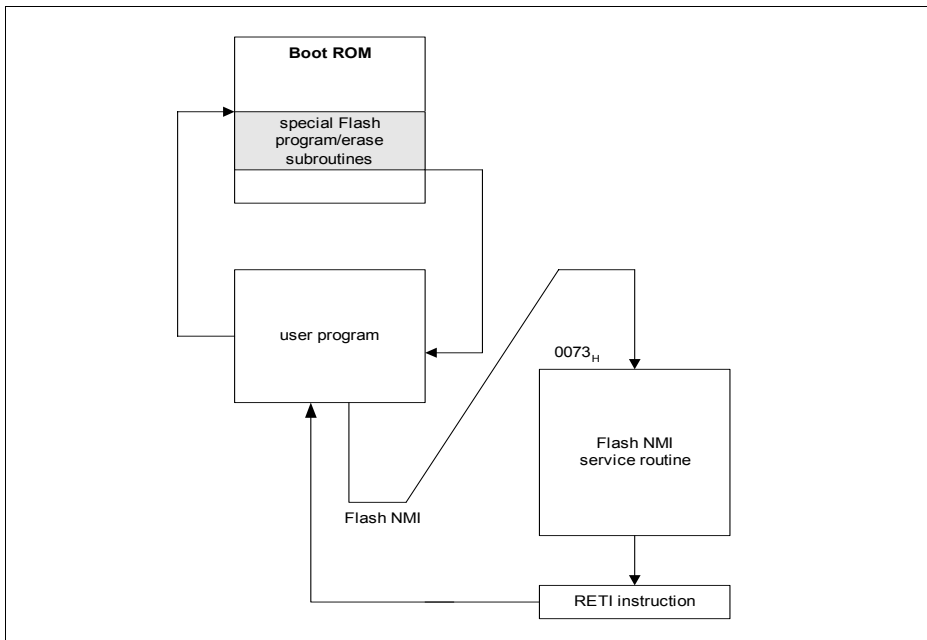


Figure 4-6 Flash Program/Erase Flow

4.7.1 Flash Programming

Each call of the Flash program subroutine allows the programming of 32 bytes of data into the selected wordline (WL) of the Flash bank.

Before calling this subroutine, the user must ensure that the 32-byte WL contents are stored incrementally in the IRAM, starting from the address specified in R5 of the current register bank. In addition, the input DPTR must contain a valid Flash WL address and all NMI flags in SFR NMISR must be 0. Otherwise, PSW.CY bit will be set and no programming will occur.

Two types of Flash program subroutines are provided in the Boot ROM to program the Flash:

- Non-background Flash Program subroutine (see [Section 4.7.1.1](#))
- Background Flash Program subroutine (see [Section 4.7.1.2](#))

4.7.1.1 Non-background Flash Program Subroutine

The first type of subroutine, Non-background Flash Program, waits until Flash programming is completed before allowing the user code to continue with its execution.

This type of routine can be used to program the Flash bank where the user code is in execution. The Flash cannot be in both program mode and read mode at the same time. It can also be used for programming the Flash bank where the interrupt vectors are defined, as interrupts cannot be handled when the Flash is in program mode.

For this subroutine, the FNMIFLASH flag in SFR NMISR is cleared automatically and therefore, need not be handled by the user.

4.7.1.2 Background Flash Program Subroutine

The second type of subroutine, Background Flash Program, allows the user program code to continue execution from where it last stopped until the Flash NMI event is generated. The FNMIFLASH flag is set and if enabled via bit NMIFLASH in SFR NMICON, an NMI to the CPU will be triggered to enter the Flash NMI service routine. At this point, the Flash bank is in ready-to-read mode i.e. programming is done.

This subroutine can be used in either of the following two cases:

- The Flash bank where the code execution is taking place, is not the same as the Flash bank that is targeted for programming.
- The subroutine is called from XRAM.

4.7.2 Flash Erasing

Each call of the Flash erase subroutine allows the user to select for erase:

- one sector; or
- a combination of sectors

Flash Memory

Before calling the Flash erase subroutine, the user must ensure that R4 to R7 of the current register bank are set accordingly. To select a sector for erase, the bit corresponding to the sector in R4 to R7 must be set to 1. Sectors not to be erased must have their corresponding bits cleared to 0. If no or invalid inputs are provided, PSW.CY bit will be set and no programming will occur.

Two types of Flash erase subroutines are provided in the Boot ROM to erase the Flash:

- Non-background Flash erase subroutine (see [Section 4.7.2.1](#))
- Background Flash erase subroutine (see [Section 4.7.2.2](#))

4.7.2.1 Non-background Flash Erase Subroutine

The first type of subroutine, non-background Flash erase, waits until erasing is completed before allowing the user code to continue with its execution.

This type of routine is necessary for users who need to erase the Flash bank where the user code is in execution. The Flash cannot be in both erase mode and read mode at the same time. It can also be used for erasing the Flash Bank where the interrupt vectors are defined, as interrupts cannot be handled when the Flash is in erase mode as the interrupt handler cannot be fetched.

For this subroutine, the FNMIFLASH flag is cleared automatically and therefore, need not be handled by the user.

Note: As program will return to user code after erasing is completed, customer should take care that they do not erase their user program code as well.

4.7.2.2 Background Flash Erase Subroutine

The second type of subroutine, background Flash erase, allows the user program code to continue execution from where it last stopped until the Flash NMI event is generated. The FNMIFLASH flag is set and if enabled via bit NMIFLASH, an NMI to the CPU will be triggered to enter the Flash NMI service routine. At this point, the Flash bank is in ready-to-read mode i.e. erasing is done.

This subroutine can be used in either of the following two cases:

- The Flash bank where the code execution is taking place, is not the same as the Flash bank that is targeted for erasing.
- The subroutine is called from XRAM.

4.7.3 Aborting Background Flash Erase

Each complete erase operation on a Flash bank requires approximately 100 ms, during which read and program operations on the Flash bank cannot be performed. For the XC82x, provision has been made to allow an on-going background Flash erase operation to be interrupted so that higher priority tasks such as reading/programming of critical data from/to the Flash bank can be performed. Hence, erase operations on

Flash Memory

selected Flash bank sector(s) may be aborted to allow data in other sectors to be read or programmed. To minimize the effect of aborted erase on the Flash data retention/cycling and to guarantee data reliability, the following points must be noted for each Flash bank:

- An erase operation cannot be aborted earlier than 5 ms after it starts.
- Maximum of two consecutive aborted erase (without complete erase in-between) are allowed on each sector.
- Complete erase operation (approximately 100 ms) is required and initiated by user-program after a single or two consecutive aborted erase as data in relevant sector(s) is corrupted.
- For the specified cycling time¹⁾, each aborted erase constitutes one program/erase cycling.
- Maximum allowable number of aborted erase for each Flash sector during lifetime is 2500.

The Flash erase abort subroutine call cannot be performed anytime within 5 ms after the erase operation has started. This is a strict requirement that must be ensured by the user. Otherwise, the erase operation cannot be aborted. Once exited from this subroutine, user can call the Flash Read Mode Status subroutine to check if the abort erase operation has been completed. When the selected bank is already in Read Mode, it indicates that the abort erase operation has been completed.

1) Refer to XC82x Data Sheet for Flash data profile

4.7.4 Flash Read Mode Status

Besides the Flash program and erase subroutines, a subroutine to check the ready-to-read mode status of the Flash Bank is additionally provided.

Before calling this subroutine, the user must ensure that the input R7 is configured to the selected Flash Bank. The carry flag is cleared when Flash is in ready-to-read mode, otherwise it is set. The carry flag is also set if the input to the subroutine is invalid.

This routine is especially useful when the abort background Flash erase subroutine is used. After calling the erase-abort routine, the user can call this Flash Read Mode Status subroutine to check if the erase operation has been successfully aborted and that the Flash is already in ready-to-read mode.

5 Boot and Startup

Entry to various boot modes such as User mode, Boot-loader mode (BSL) and On-chip Debug (OCDS) mode are done by the startup firmware in the Boot ROM. The startup firmware will depend on the Boot Mode Index (BMI) value to enter each boot mode. [Section 5.2](#) describes the behaviors of each boot modes.

The BMI/ $\overline{\text{BMI}}$ value is programmable via BSL mode 6 or user routine, BR_PROG_USER_ID. User need to program the BMI and $\overline{\text{BMI}}$ to enter each boot mode. Default mode is the UART BSL Mode. $\overline{\text{BMI}}$ is introduced to detect if the BMI value is valid. Thus, BMI value is valid when $\text{BMI} + \overline{\text{BMI}} + 1 = 0$. [Table 5-1](#) shows the BMI value required for each mode.

Table 5-1 Boot Mode Index

Boot Mode ¹⁾	BMI	$\overline{\text{BMI}}$
UART BSL Mode	00 _H	FF _H
UART BSL Mode ²⁾	00 _H	00 _H
User Mode (Productive)	10 _H	EF _H
User Mode (Diagnostic) with SPD pin SPD_0	50 _H	AF _H
User Mode (Diagnostic) with SPD pin SPD_1	52 _H	AD _H
OCDS Mode with SPD pin SPD_0	60 _H	9F _H
OCDS Mode with SPD pin SPD_1	62 _H	9D _H
Reserved	Others	Others

1) SPD : Single Pin DAP;

2) UART BSL Mode can be entered when BMI is 00 and $\overline{\text{BMI}}$ is either 00 or FF.

The BMI and $\overline{\text{BMI}}$ are part of the User Identification, USER_ID. USER_ID is a 4-byte data that contains user's specific information, see [Section 5.1](#). Beside the BMI, the user can also select either the 8 MHz active mode or 24 MHz active mode by programming the USER_ID. The mode selection is performed in the startup firmware before the start of the user code.

Once the device is in user mode (productive), changing of the BMI value to enter another boot mode could be done by a specific code embedded in the user code. This code should only be executed under a defined user condition. In this specific code, user can use one of the following instructions to change the BMI value:

- LJMP to user routine BR_UART_BSL to enter BSL mode (using BSL mode 6)
- LCALL the user routine BR_PROG_USER_ID to program USER_ID to enter other boot mode.

Boot and Startup

For code protection, user may want to erase all flash contents before executing this code. Detailed description of these user routines are found in the Boot ROM User Routines Chapter.

Note: A soft reset will be performed after the BMI has been updated. Switching off the supply voltage before the soft reset happened may cause the BMI to be updated wrongly. No additional reset is necessary for the system to use the new BMI value.

Note: It is advisable to disable the Watchdog Timer (WDT) during the changing of BMI value. A WDT reset during the programming of BMI may cause the BMI to update wrongly. Beside that, a stable power supply is also mandatory. A wrong BMI could cause the device to enter a deadlock situation.

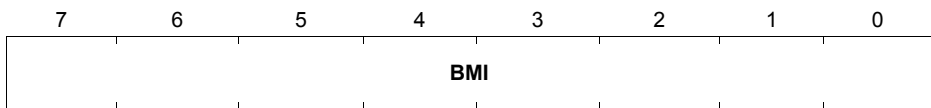
Note: Unintentional execution of the BR_PROG_USER_ID user routine will affect the BMI value. Device may not be able to boot up properly because of an invalid BMI. A user-defined condition must be included to prevent an unintentional jump to this routine.

In OCDS mode, startup firmware will based on the BMI and $\overline{\text{BMI}}$ values to enable the debug interface of the single pin DAP (SPD). In User mode (Diagnostic), the SPD interface is handled similarly as debug mode which is also based on BMI and $\overline{\text{BMI}}$ values.

5.1 User Identification Number

The USER_ID is a 4-byte data that contains user's specific information, including the boot-up options, BMI. It can be accessed by user using BR_GET_4_BYTES_INFO user routine or via BSL Mode A. USER_ID can be programmed via BR_FEATURE_SETTING user routine or via BSL Mode 6. Detailed description of these BSL Modes can be found in UART Boot-loader chapter.

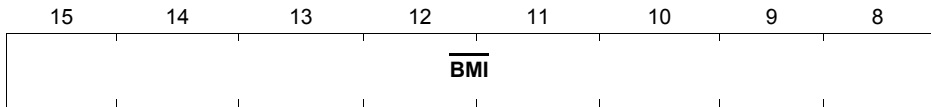
USER_ID, Byte 0



Field	Bits	Description
BMI	[7:0]	Boot Mode Index The boot option to enter User Mode, UART BSL Mode and OCDS Mode as described in Table 5-1 .

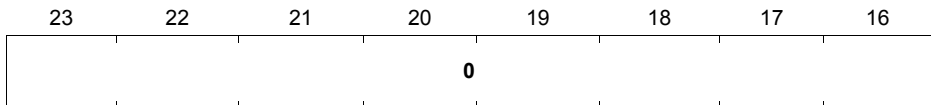
Boot and Startup

USER_ID, Byte 1



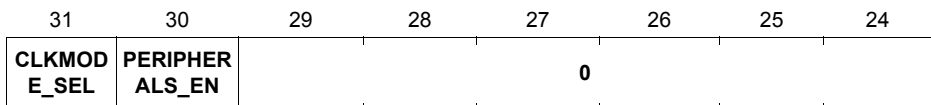
Field	Bits	Description
BMI	[15:8]	Inverse of Boot Mode Index The inverse of BMI. BMI is checked for validity with $\overline{\text{BMI}}$. BMI is valid when $\text{BMI} + \overline{\text{BMI}} + 1 = 0$

USER_ID, Byte 2



Field	Bits	Description
0	[23:16]	Reserved

USER_ID, Byte 3



Field	Bits	Description
0	[29:24]	Reserved

Boot and Startup

Field	Bits	Description
PERIPHERALS_EN	30	Peripherals Enable Bit 0 Disable all peripherals defined in register PMCON1 1 Enable all peripherals defined in register PMCON1
CLKMODE_SEL	31	Clock Mode Selection 0 8 MHz active mode 1 24 MHz active mode

5.2 Boot ROM Operating Mode

After a reset, the CPU will always start by executing the Boot ROM code which occupies the program memory address space $0000_H - 1FFF_H$. The Boot ROM start-up procedure will first switch the address space for the Boot ROM to $C000_H - DFFF_H$. Then remaining Boot ROM start-up procedure will be executed from $C00X_H$. This includes checking the programmed BMI value to enter the selected Boot ROM operating modes. The memory organization of the XC82x shown in this document is after the Boot ROM address switch where the different operating modes are executed.

5.2.1 User Mode (Productive)

If the User mode (productive) is selected, the Boot ROM will jump to program memory address 0000_H to execute the user code in the Flash memory. In this mode, the content in the Flash memory are protected from external access. This is the normal operating mode of the XC82x.

5.2.2 User Mode (Diagnostic)

If the User mode (diagnostic) is selected, the Boot ROM will jump to program memory address 0000_H to execute the user code in the Flash memory. This is similar to the user mode (productive) described in [Section 5.2.1](#), with the addition that the specified SPD port is automatically configured to allow hot-attach.

5.2.3 Boot-Loader Mode

If the Boot-loader (BSL) mode is selected, the software routines of the BSL located in the Boot ROM will be executed, allowing the XRAM and Flash memory to be programmed, erased and executed. Refer to the UART BSL chapter for the different BSL working modes.

5.2.4 OCDS Mode

If the OCDS mode is selected, the debug mode will be entered for debugging program code. The OCDS hardware is initialized and a jump to program memory address 0000_H is performed next. The user code in the Flash memory is executed and the debugging process may be started.

During the OCDS mode, the lowest 64 bytes (00_H – 3F_H) in the internal data memory address range may be alternatively mapped to the 64-byte monitor RAM or the internal data RAM.

6 UART Boot-Loader

The XC82x includes a UART Boot-Loader (BSL) Mode that can be entered with the BMI settings as described in Boot and Startup chapter. The main purpose of BSL Mode is to allow easy and quick programming/erasing of the Flash and XRAM via UART.

The UART BSL mode consists of two functional parts that presents two phases as described below:

- **Phase I:** Establish a serial connection and automatically synchronize to the transfer speed (baud rate) of the serial communication partner (host).
- **Phase II:** Perform the serial communication with the host. The host controls the communication by sending special header information which selects one of the working modes. These modes are:
 - **Mode0: Program customer code to XRAM**
 - **Mode1: Execute customer code in XRAM**
 - **Mode2: Program customer code to FLASH**
 - **Mode3: Execute customer code in FLASH**
 - **Mode4: Erase customer code in FLASH sector(s)**
 - **Mode6: Program 4 bytes of USER_ID**
 - **ModeA: Get 4 bytes Information**

Except **Mode 1**, **Mode 3** and **Mode 6**, the μ C would return to the beginning of Phase II and wait for the next command from the host after executing all other Modes.

The serial communication, which is activated in Phase II, is based a byte received in Phase I (after baud rate 0x0055) to select **single pin (0xAA)** or **dual pins (0x55)** UART of XC82x. UART auto-baud routine uses the LIN baud rate detection to calculate the baud rate. The port settings for BSL Mode are listed in [Table 6-1](#).

Table 6-1 Port Settings for UART BSL Mode

Device	UART_INIT_ID	Single / Dual Pin(s)	Pin(s) used
XC82x	0xAA	Single pin	P 0.6 as RXD P 0.6 as TXD
XC82x	0x55	Dual pins	P 0.6 as RXD P 0.5 as TXD

The serial transfer is working in asynchronous mode with the serial parameters 8N1 (eight data bits, no parity and one stop bit). The host can vary the baud rate in a wide range because the μ C does an automatic synchronization with the host in Phase I.

6.1 Phase I: Automatic Serial Synchronization to the Host

Upon entering UART BSL Mode, a serial connection is established and the transfer speed (baud rate) of the serial communication partner (host) is automatically synchronized in the following steps:

- STEP 1: Initialize serial interface for reception and timer for baud rate measurement
- STEP 2: Wait for SYN Break (13 bits low) and SYN Byte (0x55) from host
- STEP 3: Synchronize the baud rate to the host
- STEP 4: Acquire UART_INIT_ID (0xAA or 0x55) to determine single pin or dual pins UART and initialize pins
- STEP 5: Send Acknowledge byte (55H) to the host
- STEP 6: Enter Phase II

6.1.1 General Description

The baud rate detection feature provides the capability to detect the baud rate using Timer 2. Initialization consists of:

- Serial port of the microcontroller set to Mode 1 (8-bit UART, variable baud rate) for communication
- Baud rate selection for detection (BCON.BGSEL) is defined as 00 (left as default)
- Capture Timer 2 data register contents on negative transition at pin T2EX
- Timer 2 external events are enabled (EXF2 flag is set when a negative transition occurs at pin T2EX)
- $f_{T2} = f_{PCLK} / 4$ (T2PRE=010)

The auto baud for UART BSL consists of the:

- SYN Break (13 Bit time low)
- SYN byte (55_H)
- UART_INIT_ID

The Break is used to signal the beginning of a new frame and must be at least 13 bits of dominant value. When negative transition is detected at pin T2EX at the beginning of Break, the Timer 2 External Start Enable bit (T2MOD.T2RHEN) is set. This will then automatically start Timer 2 at the next negative transition of pin T2EX. Finally, the End of SYN Byte Flag (FDCON.EOFSYN) is polled. When this flag is set, Timer 2 is stopped. T2 Reload/Capture register (RC2H/L) is the time taken for 8 bits. Then the auto baud routine calculates the actual baud rate, sets the PRE and BG values and activates Baud Rate Generator. Once all are done, UART_INIT_ID value is read to determine the setting of the UART BSL Mode to be single or dual pins, before ACK bytes (0x55) are sent out. The baud rate detection is shown in [Figure 6-1](#)

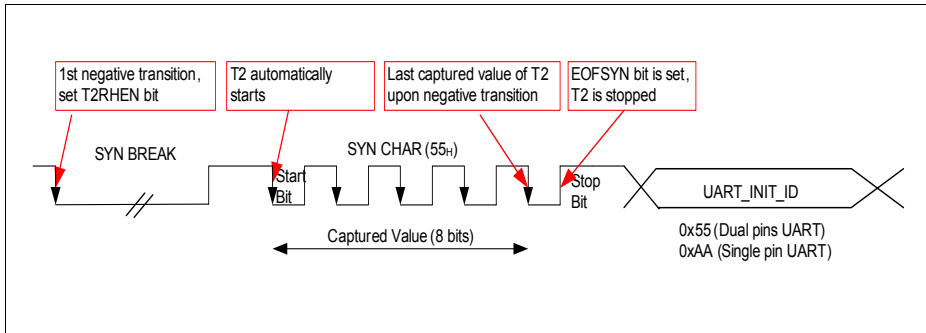


Figure 6-1 Auto Baud Rate Detection

6.1.2 Calculation of BG and PRE values

To set up auto baud rate detection, the BG and PRE must be preload. As there are two unknown values, two formulas are therefore needed. Firstly, the correlation between the baud rate (baud) and the reload value (BG) depends on the internal peripheral frequency (f_{PCLK}):

$$\text{baud} = \frac{f_{PCLK}}{16 \left(\text{PRE} \times \left(\text{BRVALUE} + \frac{n}{32} \right) \right)} \quad (6.1)$$

Second, the relation between the baud rate (baud) and the recording value of Timer 2 (T_2) depends on the T_2 peripheral frequency (f_{T_2}) and the number of received bits (N_b)

$$\text{baud} = \frac{f_{T_2} \times N_b}{T_2} \quad (6.2)$$

Combining [Equation \(6.1\)](#) and [Equation \(6.2\)](#), together with $N_b = 8$, $f_{T_2} = f_{PCLK} / 4$ ($T2PRE=010$), $PRE=1$

$$\frac{f_{PCLK}}{16 \left(\text{PRE} \times \left(\text{BRVALUE} + \frac{n}{32} \right) \right)} = \frac{\frac{f_{PCLK}}{4} \times 8}{T_2} \quad (6.3)$$

Simplifying [Equation \(6.3\)](#), we get

(6.4)

$$\left(\text{BRVALUE} + \frac{n}{32} \right) = \frac{T2}{32}$$

The capturing of the timing for 8 bits makes the formula easy for realization in assembly language. The division with 32 can be simply achieved by a 5-bit right shift operation. In previous Acropolis family, the shift operation causes a loss in the decimal digits, thus reducing the baud rate accuracy. Therefore in XC82x, the concept (SFRs BGL and BGH) is implemented in such a way that the actual shift (division) is not necessary by firmware as it will be executed by hardware. In this way, we keep the decimal digits after division and giving a better accuracy of the baud rate detection. Thus after capturing the 8 bits timing in R2CH and R2CL, the values are programmed directly to BGH and BGL respectively.

After setting BG and PRE, the Baud Rate Generator will then be enabled, and the subsequent frames will follow this baud rate. After receiving UART_INIT_ID to determine single or dual pins for UART BSL, the UART BSL routine sends an Acknowledge byte (55_H) to the host. If this byte is received correctly, it will be guaranteed that both serial interfaces are working with the same baud rate.

6.2 Phase II: Serial Communication Protocol and the Modes

After the successful synchronization to the host, the UART BSL routine enters Phase II, during which it communicates with the host to select the desired working modes. The detailed communication protocol is explained as follows:

6.2.1 Serial Communication Protocol

The communication between the host and the UART BSL routine is done by a simple transfer protocol. The information is sent from the host to the μC in blocks. All the blocks follow the specified block structure. The communication is nearly unidirectional, that is, that the host is sending several transfer blocks and the UART BSL routine is just confirming them by sending back single acknowledge or error bytes. The μC itself does not send any transfer blocks.

However, the above regulation does not apply to some modes where the μC might need to send the required data to the host besides the acknowledge or error byte.

6.2.1.1 Transfer Block Structure

A transfer block consists of three parts:

UART Boot-Loader

Block Type (1 byte)	Data Area (X bytes)	Checksum (1 byte)
------------------------	------------------------	----------------------

- **Block Type:** the type of block, which determines how the data in the **data area** is interpreted. Implemented block types are:
 - 00_H type “HEADER”
 - 01_H type “DATA”
 - 02_H type “END OF TRANSMISSION” (EOT)
- **Data Area:** A list of bytes, which represents the data of the block. The length of **data area** cannot exceed 96/97 bytes for Mode 0 and 2, depending on whether it is a Data Block or EOT Block
- **Checksum:** the XOR checksum of the **block type** and **data area**.

The host will decide the number of transfer blocks and their respective lengths during one serial communication process. For safety purpose, the last byte of each transfer block is a simple **checksum** of the **block type** and **data area**. The host generates the checksum by **XOR-ING** all the bytes of the **block type** and **data area**. Every time the UART BSL routine receives a transfer block, it recalculates the checksum of the received bytes (**block type** and **data area**) and compares it with the attached checksum.

Note: If there is less than 1WL to be programmed to Flash, the PC Host will have to fill up the vacancies with 00_H, and transfer data in the length of 32n bytes (n=1-3).

6.2.1.2 Transfer Block Type

There are three types of transfer blocks depending on the value of the **block type**. **Table 6-2** provides the general information on these block types. More details will be described in the corresponding sections later.

Table 6-2 Type of Transfer Block

Block Name	Block Type	Description
Header Block	00 _H	This block has a fixed length of 8 bytes. Special information is contained in the data area of the block, which is used to select different working modes.

UART Boot-Loader

Table 6-2 Type of Transfer Block

Block Name	Block Type	Description
Data Block	01 _H	This block length depends on the special information given in the previous header block. This block is used in working mode 0 and 2 to transfer a portion of program code. The program code is in the data area of the block.
EOT Block	02 _H	This block length depends on the special information given in the previous header block. This block is the last block in data transmission in working mode 0 and 2. The last program code to be transferred are in the data area of the block.

6.2.1.3 Response codes to the host

The μ C would let the host know whether a block has been successfully received by sending out a response code. If a block is received correctly, an Acknowledge Code (55_H) is sent. In case of failure, there are two kinds of errors. The error might be a wrong block type and the UART BSL would send back a block type error (0FF_H) to the host. This kind of error is caused by two conditions. One is the μ C receives a block type other than the implemented ones. The other is the μ C receives the transfer blocks in wrong sequences. For example, in working mode 0, immediately after the header block is received, if another header block instead of a data block is received, the μ C would consider this case be a wrong block type error. Besides wrong block type error, the other error is checksum error. If the checksum comparison fails, the UART BSL routine is rejecting the transfer block by sending back a checksum error code (0FE_H) to the host. In both error cases the UART BSL routine awaits the actual block from the host again.

There is no flash protection, thus the flash protection type error is removed for XC82x. UART BSL does not check for validity of address of XRAM/Flash, it is to the user's responsibility to ensure that the DPTR of Flash/XRAM is valid.

Table 6-3 gives a summary of the response codes to be sent back to the host by the μ C after it receives each transfer block.

Table 6-3 Type of Response Codes

Communication Status	Response Code to the Host
Successful	55 _H
Block Type Error	0FF _H
Checksum Error	0FE _H

Table 6-4 shows a tabulated summary of the possible responses the device may transmit following the reception of a Header, Data or EOT Block.

Table 6-4 Possible Responses for Various Block Types

Mode	Header Block	Data Block	EOT Block
0	Acknowledge, Block Type Error, Checksum Error	Acknowledge, Block Type Error, Checksum Error	Acknowledge, Block Type Error, Checksum Error
1	Acknowledge, Block Type Error, Checksum Error		
2	Acknowledge, Block Type Error, Checksum Error	Acknowledge, Block Type Error, Checksum Error	Acknowledge, Block Type Error, Checksum Error
3	Acknowledge, Block Type Error, Checksum Error		
4	Acknowledge, Block Type Error, Checksum Error		
6	Acknowledge, Block Type Error, Checksum Error		
A	Acknowledge, Block Type Error, Checksum Error		
E	Acknowledge, Block Type Error, Checksum Error		

The responses are defined in [Table 6-5](#), which lists the possible reasons and/or implications for error and suggests the possible corrective actions that the host can take upon notification of the error.

Table 6-5 Definitions of Responses

Response	Value	Description			
		Block Type	BSL Mode	Reasons / Implications	Corrective Action
Acknowledge	55 _H	Header	1, 3	The requested operation will be performed once the response is sent.	
			6, A	The requested operation has been performed and is successful.	
		EOT	0, 2, 4		
		All others		Reception of the Block is successful. Transmission of IDs follow in Mode A. Ready to receive the next Block.	
Block Error	FF _H	All others		Either the Block_Type is undefined or option is invalid or the flow is invalid.	Retransmit a valid Block
Checksum Error	FE _H	All		There is a mismatch between the calculated and the received Checksum.	Retransmit a valid Block

6.2.2 The Selection of Working Modes

When the UART BSL routine enters Phase II, it first waits for an eight-byte long header block from the host. The header block contains the information for the selection of the working modes. Depending on this information, the UART BSL routine selects and activates the desired working mode. If the μ C receives an incorrect header block, the UART BSL routine sends, instead of an Acknowledge code, a checksum or block error code to the host and awaits the header block again. In this case the host may react by re-sending the header block or by releasing a message to the customer.

6.2.2.1 Receiving the Header Block

The header block is always the first transfer block to be sent by the host during one data communication process. It contains the working mode number and special information on the related mode (referred to as "mode data"). The general structure of a Header Block is shown below.

Block Type 00 _H (Header Block)	Data Area		Checksum (1 byte)
	Mode (1 byte)	Mode Data (5 bytes)	

Description:

- **00_H**: The block type, which marks the block as a **Header Block**
- **Mode**: The working mode. The implemented working modes are:
 - **00_H** (**Mode0: Program customer code to XRAM**)
 - **01_H** (**Mode1: Execute customer code in XRAM**)
 - **02_H** (**Mode2: Program customer code to FLASH**)
 - **03_H** (**Mode3: Execute customer code in FLASH**)
 - **04_H** (**Mode4: Erase customer code in FLASH sector(s)**)
 - **06_H** (**Mode6: Program 4 bytes of USER_ID**)
 - **0A_H** (**ModeA: Get 4 bytes Information**)
- **Mode Data**: Five bytes of special information, which are necessary to activate corresponding working mode.
- **Checksum**: The checksum of the header block.

6.2.2.2 Mode0: Program customer code to XRAM

Mode 0 is used to transfer a customer program from the host to the XRAM of the μ C via serial interface.

The header block for this working mode has the following structure:

The Header Block

00 _H (Header Block)	00 _H (Mode 0)	Data Area				Checksum (1 byte)
		StartAddr High (1 byte)	StartAddr Low (1 byte)	Block Length (1 byte)	Not Used (2 bytes)	

Mode Data Description:

Start Addr High, Low: 16-bit Start Address, which determines where to copy the received program codes in the XRAM.

Block_Length: The length of the following Data Blocks or EOT Block at each transmission.

Not used: 2 bytes, these bytes are not used and will be ignored in Mode 0.

UART Boot-Loader

*Note: The **Block-Length** refers to the whole length (block type, data area and checksum) of the following transfer block (Data Block or EOT Block). For each transmission, length of Data Block(s) and EOT Block should be the same.*

Note: Maximum length for Data Blocks (when sending Header Block, followed by 1-255 Data Blocks and finally end transmission with one EOT Block) is $96+2 = 98$ bytes.

Note: Maximum length for EOT Blocks (when sending one Header Block followed by one EOT Block each time only) is $96 + 3 = 99$ bytes

After successfully receiving the Header Block, the μ C enters Mode 0, during which the program codes are transmitted from the host to the μ C by Data Block(s) and EOT Block, which are describe as below.

Note: No empty Data Block is allowed.

The Data Block

01_H (Data Block) (1 byte)	Program Codes ((Block_Length-2) bytes)	Checksum (1 byte)
--	---	-----------------------------

Description:

Program Codes: The program codes have a length of (**Block_Length-2**) byte, where the **Block_Length** is provided in the previous Header Block.

The EOT Block

02_H (EOT Block) (1 byte)	Last_Code_Length (1 byte)	Program Code	Not Used	Checksum (1 byte)
---	-------------------------------------	---------------------	-----------------	-----------------------------

Description:

Last_Codelength: This byte indicates the length of the program codes in this EOT Block.

Program Codes: The last program codes to be sent to the μ C

Not used: The length is (**Block_Length-3-Last_Codelength**). These bytes are not used and they can be set to any value.

6.2.2.3 Mode1: Execute customer code in XRAM

Mode 1 is used to execute a customer program in the XRAM of the μC at 0F000_H. The Header Block for this working mode has the following structure:

The Header Block

00 _H (Header Block)	01 _H (Mode 1)	Data Area	Checksum (1 byte)
		Not Used	

Mode Data Description:

Not used: The five bytes are not used and will be ignored in Mode 1.

Under working Mode 1, the Header Block is the only transfer block to be sent by the host, no further serial communication is necessary. The μC will exit the UART BSL Mode and jump to the XRAM address at 0F000_H.

6.2.2.4 Mode2: Program customer code to FLASH

Mode 2 is used to transfer a customer program from the host to the Flash of the μC via serial interface.

The header block for this working mode has the following structure:

The Header Block

00 _H (Header Block)	02 _H (Mode 2)	Data Area				Checksum (1 byte)
		StartAddr High (1 byte)	StartAddr Low (1 byte)	Block Length (1 byte)	Not Used (2 bytes)	

Mode Data Description:

Start Addr High, Low: 16-bit Start Address, which determines where to copy the received program codes in the Flash. This address must be valid and aligned to the page address.

Block_Length: The length of the following Data Blocks or EOT Block. At each time, PC Host can sent minimum 1WL (32 bytes) and maximum 3 WLs (96 bytes). If data blocks are to be sent, the maximum block_length has to be 98 (=96+2) bytes. If only EOT block is sent, the maximum block_length has to be 99 (=96+3)bytes.

Not used: 2 bytes, these bytes are not used and will be ignored in Mode 2.

UART Boot-Loader

*Note: If the data starts in a non-page address, PC Host must fill up the beginning vacancies with 00_H and provide the **start address** of that page address. For e.g., if data starts in 0F82_H, the PC Host will fill up the addresses 0F80_H and 0F81_H with 00_H and provide the **Start Address** 0F80_H to microcontroller. And if data is only 8 bytes, the PC Host will also fill up the remaining addresses with 00_H and transfer 32n data bytes.*

*Note: The **Block_Length** refers to the whole length (block type, data area and checksum) of the following transfer block (Data Block or EOT Block). Since the data area is multiples of 32 bytes, thus by adding block type and checksum bytes, the **Block_Length** is 32n+2 bytes.*

After successfully receiving the Header Block, the μ C enters Mode 2, during which the program codes are transmitted from the host to the μ C by Data Block and EOT Block, which are describe as below.

Note: No empty Data Block is allowed.

The Data Block

01 _H (Data Block) (1 byte)	Program Codes ((Block_Length-2) bytes)	Checksum (1 byte)
---	---	----------------------

Description:

Program Codes: The program codes have a length of (Block_Length-2) byte, where the Block_Length is provided in the previous Header Block.

The EOT Block

02 _H (EOT Block) (1 byte)	Last_Code_Length (1 byte)	Program Code	Not Used	Checksum (1 byte)
--	------------------------------	--------------	----------	----------------------

Description:

Last_Codelength: This byte indicates the length of the program codes in this EOT Block.

Note: If Data blocks are sent, this byte should be zero. If this byte is not zero, additional undesired bytes will be programmed

Program Codes: The last program codes to be sent to the μ C

UART Boot-Loader

Not used: The length is (**Block_Length-3-Last_Codelength**) and should be filled with zeros.

6.2.2.5 Mode3: Execute customer code in FLASH

Mode 3 is used to execute a customer program in the Flash of the μC at 0000_H. The Header Block for this working mode has the following structure:

The Header Block

00 _H (Header Block)	03 _H (Mode 3)	Data Area	Checksum (1 byte)
		Not Used	

Mode Data Description:

Not used: The five bytes are not used and will be ignored in Mode 3.

In working Mode 3, the Header Block is the only transfer block to be sent by the host, no further serial communication is necessary. The μC will exit the UART BSL Mode and jump to the Flash address at 0000_H.

6.2.2.6 Mode4: Erase customer code in FLASH sector(s)

Mode 4 is used to erase different sector in the Flash Bank 0.

The Header Block for Flash Sector erase

00 _H (Header Block)	04 _H (Mode 4)	Data Area			Checksum (1 byte)
		Flash_B0_ SectorL (1 byte)	Flash_B0_ SectorH (1 byte)	Not Used (3 bytes)	

Mode Data Description:

Flash_B0_SectorL: In this byte, the sectors 0 to 7 of Flash Bank 0 are represented by bits 0 to 7. When the bit contains a 1, the corresponding sector is selected. E.g. byte of 0x12 selects sectors 1 and 4 of Flash Bank 0 for erase.

Flash_B0_SectorH: In this byte, the sectors 8 to 9 of Flash Bank 0 are represented by bits 0 to 1. When the bit contains a 1, the corresponding sector is selected. E.g. byte of 0x01 selects sectors 8 of Flash Bank 0 for erase.

Note: Unwanted/unselected bits should be cleared to 0

UART Boot-Loader

6.2.2.7 Mode6: Program 4 bytes of USER_ID

Mode 6 is used to program User identification, USER_ID (4 bytes). The Header block for this working mode has the following structure:

The Header Block

00 _H (Header Block)	06 _H (Mode 6)	Data Area					Checksum (1 byte)
		USER_ID_ 3 (1 byte)	USER_ID_ 2 (1 byte)	USER_ID_ 1 (1 byte)	USER_ID_ 0 (1 byte)	Not Used (1 byte)	

Mode Data Description

User_ID: These 4 bytes is given by user as USER_ID for BMI values and initialization of settings.

Not used: This byte is not used and will be ignored in Mode 6.

After programming the USER_ID and sending back the acknowledge response to the Host (to indicate successful operation has executed), a **software reset** will be triggered and the programmed boot-up mode will be entered.

Note: This Mode should be handled and executed with care. It should be executed last. User should program their Flash code first, program XRAM code (if needed), and finally program USER_ID.

Note: There is no erasing option for USER_ID. To erase the USER_ID, simply write 0x00 to the 4 bytes USER_ID.

6.2.2.8 ModeA: Get 4 bytes Information

Mode A is used to get 4 bytes data determined by the Option byte in the header block. The header block for this working mode has the following structure:

The Header Block

00 _H (Header Block)	0A _H (Mode A)	Data Area		Checksum (1 byte)
		Not Used (4 bytes)	Option (1 byte)	

Mode Data Description:

UART Boot-Loader

Option: This byte will determine the 4 bytes data to be sent to the host. Only option 00_H - 02_H are valid options.

00_H - Chip Identification Number (MSB byte 1... LSB byte 4)

01_H - USER_ID (MSB byte 1... LSB byte 4)

In Mode A, the header block is the only transfer block to be sent by the host. The microcontroller will return an acknowledgement followed by 4 bytes of data to the host if the header block is received successfully. If an invalid option is received, the microcontroller will return 4 bytes of 00_H. USER_ID is the user identification number and the order of the 4 bytes of USER_ID_INFO are as followed: USER_ID_4, USER_ID_3, USER_ID_2 and USER_ID_1.

7 System Control Unit

The System Control Unit (SCU) of the XC82x handles all system control tasks besides the debug related tasks which are controlled by the OCDS/Cerberus. All functions described in this chapter are tightly coupled, thus, they are conveniently handled by one unit, the SCU. The SCU contains the following functional sub-blocks:

- Embedded Voltage Regulator (EVR) (see [Section 7.1 on Page 7-1](#))
- Reset Control (see [Section 7.2 on Page 7-6](#))
- Clock System and Control (see [Section 7.3 on Page 7-11](#))
- Power Management (see [Section 7.4 on Page 7-16](#))

The XC82x provides a range of utility features for secure system performance under critical conditions (e.g., brownout).

At the center of the XC82x clock system is the Clock Control Unit (CCU), which generates a master clock frequency using the 48 MHz oscillator. In-phase synchronized clock signals are derived from the master clock and distributed throughout the system.

7.1 Power Supply System with Embedded Voltage Regulator

The power supply to the core, memories and the peripherals is regulated by the Embedded Voltage Regulator (EVR) that comes with detection circuitries to ensure that the supplied voltages are within the specified operating range. The main voltage and low power voltage regulators in the EVR may be independently switched off to reduce power consumption for the different power saving modes.

The XC82x microcontroller requires two different levels of power supply:

- 2.5 V to 5.5 V for the Embedded Voltage Regulator (EVR) and Ports
- 2.5 V for the core, memory, on-chip oscillator, and peripherals

Figure 7-1 shows the XC82x power supply system. A power supply of 2.5 V to 5.5 V must be provided from the external power supply pin. The 2.5 V power supply for the logic is generated by the EVR. The EVR helps reduce the power consumption of the whole chip and the complexity of the application board design.

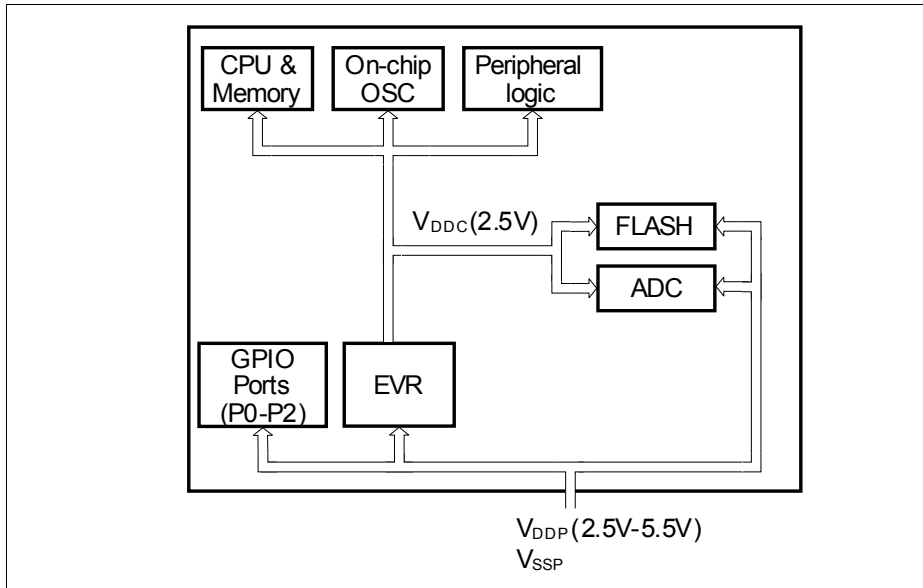


Figure 7-1 XC82x Power Supply System

EVR Features:

- Input voltage (V_{DDP}): 3.0 V to 5.5 V @ full operation condition
- Input voltage (V_{DDP}) down to 2.5 V in active mode or power down mode with a reduction in the load conditions @ reduced voltage condition
- Output Voltage (V_{DDC}): 2.5 V +/- 7.5%
- Low power voltage regulator provided in power-down mode
- V_{DDC} and V_{DDP} prewarning detection
- V_{DDC} brownout detection

The EVR consists of a main voltage regulator and a low power voltage regulator. In active mode, both voltage regulators are enabled. In power-down mode, the main voltage regulator is switched off, while the low power voltage regulator continues to function and provide power supply to the system with low power consumption.

The EVR works within an input voltage range of 3.0 V to 5.5V under full operation condition and within an input range down to 2.5 V under reduced voltage condition. XC82x will only work in power down mode or in active mode with limited load available in the reduced voltage condition.

The EVR has the V_{DDC} and V_{DDP} detectors. There are two threshold voltage levels for V_{DDC} detection: prewarning (2.4 V) and brownout (2.3 V). When V_{DDC} is below a nominal voltage of 2.4 V, the V_{DDC} NMI flag, NMISR.FNMIVDDC is set and an NMI request to the

System Control Unit

CPU is activated provided V_{DDC} NMI is enabled (NMICON.NMIVDDC). The prewarning detection is disabled by default via bit VDDCPW in SDICON register. This is necessary especially in the reduced voltage condition. This bit has to be set to 1 to enable the prewarning detection. If V_{DDC} is below a nominal voltage of 2.3 V, the brownout reset will be activated, putting the microcontroller into a reset state. In power down mode, brownout reset happens when V_{DDC} is below 1.5 V.

For V_{DDP} , there is a nominal prewarning level of 3.6 V and a nominal brownout reset threshold of 2.9 V. When V_{DDP} is below a nominal voltage of 3.6 V, the V_{DDP} NMI flag, NMISR.FNMIVDDP is set and an NMI request to the CPU is activated provided V_{DDP} NMI is enabled (NMICON.NMIVDDP). If V_{DDP} is below a nominal voltage of 2.9 V, the brownout reset will be activated, putting the microcontroller into a reset state. The detection of these 2 levels could be disabled via bits VDDPPW and VDDPBOA in SDICON register in active mode especially in the reduced voltage condition. These 2 detections are automatically shut down in power down mode. In power down mode, brownout reset happens when V_{DDP} is below 3.6 V if bit VDDPBOPD is set to 1.

Besides the various brownout threshold levels for V_{DDP} and V_{DDC} , EVR enters into the reset mode when V_{DDP} is below 2.4 V. It could happened if the V_{DDP} brownout detection in active mode or power down mode is disabled via bit VDDPBOA and VDDPBOPD.

7.1.1 Reduced Voltage Condition

In the reduced voltage condition of $2.5\text{ V} < V_{DDP} < 3.0\text{ V}$, the active current must be below a sets of values based on the EVR driving capability. These limits can be found in the Data Sheet. The active current consumption needs to be below the specified values as according to the V_{DDP} voltage. If the conditions are not met, a brownout reset may be triggered. A guideline of the current consumption for some of the modules is also available in the datasheet.

Note: The full operation of XC82x is specified for $3\text{ V} < V_{DDP} < 5.5\text{ V}$.

System Control Unit

7.1.2 EVR Register Description

The SDCON is used to enable or disable the various detectors. The status of V_{DDP} and V_{DDC} threshold levels are also indicated in this register. The bit field PAGE of SCU_PAGE register must be programmed before accessing these registers.

SDCON

Supply Detection Control Register (EE_H)

Reset Value: 34_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	VDDPTH	VDDCTH	VDDPBO PD	VDDPBOA	VDDPPW	VDDCPW	
r	rh	rh	rw	rw	rw	rw	

Field	Bits	Type	Description
VDDCPW	0	rw	V_{DDC} Prewarning Detection Enable 0 V_{DDC} prewarning detection is disabled. 1 V_{DDC} prewarning detection is enabled. <i>Note: VDDC prewarning flag and NMI will only be triggered when this bit is set to 1.</i>
VDDPPW	1	rw	V_{DDP} Prewarning Detection Enable 0 V_{DDP} prewarning detection is disabled. 1 V_{DDP} prewarning detection is enabled. <i>Note: VDDP prewarning flag and NMI will only be triggered when this bit is set to 1.</i>
VDDPBOA	2	rw	V_{DDP} Brownout Detection Enable in active mode 0 V_{DDP} brownout detection in active mode is disabled. 1 V_{DDP} brownout detection in active mode is enabled.
VDDPBOPD	3	rw	V_{DDP} Brownout Detection Enable in power down mode 0 V_{DDP} brownout detection in power down mode is disabled. 1 V_{DDP} brownout detection in power down mode is enabled.

System Control Unit

Field	Bits	Type	Description
VDDCTH	4	rh	V_{DDC} Threshold Indication 0 Below V _{DDC} prewarning threshold level. 1 Above V _{DDC} prewarning threshold level. <i>Note: It is not affected by the bit VDDCPW.</i>
VDDPTH	5	rh	V_{DDP} Threshold Indication 0 Below V _{DDP} prewarning threshold level. 1 Above V _{DDP} prewarning threshold level. <i>Note: It is not affected by the bit VDDPPW.</i>
0	[7:6]	r	Reserved Returns 0 if read; should be written with 0.

7.2 Reset Control

The XC82x has five types of resets: power-on reset, watchdog timer reset, soft reset, power-down wake-up reset, and brownout reset.

When the XC82x is first powered up or with brownout condition triggered by supply voltage input(s) going below the threshold, proper voltage thresholds must be reached before the system starts operation with the release of the system reset. The CPU then starts to execute from the Boot ROM firmware.

The Watchdog Timer (WDT) module is also capable of resetting the device if it detects a malfunction in the system.

Another type of reset that needs to be detected is the reset while the device is in power-down mode (i.e., wake-up reset). While the contents of the static RAM are undefined after a power-on reset, they are well defined after a wake-up reset from power-down mode.

A brownout reset is triggered if the V_{DDC} supply voltage dips below 2.3 V or V_{DDP} supply voltage dips below 2.9 V provided the detector is enabled.

As for soft reset, it can be triggered by application software where applicable.

7.2.1 Types of Reset

7.2.1.1 Power-On Reset

The supply voltage V_{DDP} is used to power up the chip. The EVR is the first module in the chip to be reset, which includes:

1. Startup of the main voltage regulator and the low power voltage regulator.
2. When V_{DDP} and V_{DDC} reach the threshold of the V_{DDP} and V_{DDC} detectors, the reset of EVR becomes inactive.

When the system starts up, the device is running in active 8 MHz mode using internal 48 MHz oscillator clock as the system frequency. Once the 48 MHz oscillator is stable which is 480 oscillations after EVR is stable, Flash has to be in the ready-to-read mode before the deassertion of the system reset. In user mode, the system clock can be switched by the startup firmware in Boot ROM to the user selectable mode (active 8 MHz mode or 24 MHz mode). P0.4 pin serves as a reset indication to the external world to indicate that the device has executed a reset.

The startup firmware starts to run after the system is out of reset. The user specified settings in USER_ID such as the boot mode (User Mode, BSL Mode, OCDS Mode) to enter and the system clock speed will be decoded in the startup firmware. Based on the value decoded, the startup firmware will performed the necessary setup before entering the selected boot mode.

System Control Unit

Note: When V_{DDP} is not powered on, the current over any GPIO pin must not source V_{DDP} higher than 0.3 - 0.5 V.

7.2.1.2 Watchdog Timer Reset

The watchdog timer reset is an internal reset. The Watchdog Timer (WDT) maintains a counter that must be refreshed or cleared periodically. If the WDT is not serviced correctly and in time, it will generate an NMI request to the CPU and then reset the device after a predefined time-out period. Bit PMCON0.WDTRST is used to indicate the watchdog timer reset status.

For watchdog timer reset, as the EVR and 48 MHz oscillator is already stable, the timing for watchdog timer reset is shorter compared to the other types of resets.

7.2.1.3 Soft Reset

Soft reset is an internal reset that is caused by a software set of the soft reset request bit, SWRQ, in RSTCON register. It can be triggered by application software where necessary.

7.2.1.4 Power-Down Wake-Up Reset

Power is still applied to the XC82x during power-down mode, as the low power voltage regulator is still operating. If power-down mode is entered appropriately, all important system states will have been preserved in the Flash by software.

If the XC82x is in power-down mode, three options are available to awaken it:

- through RTC wake-up event (depending on the type of power down mode)
- through EXINT0 pin
- through the failure of the RTC clock source

Selection of option through EXINT0 pin is made via the control bit PMCON0.EWS. The RTC wake-up event and RTC clock failure could be used as the wake-up sources depending on the type of power down mode. The wake-up from power-down can be with reset or without reset; this is chosen by the PMCON0.WKSEL bit. The wake-up status (with or without reset) is indicated by the RSTCON.WKRS bit.

The time needed for the device to be out of reset is faster as compared to the power-on reset as the EVR takes a shorter time period to become stable.

7.2.1.5 Brownout Reset

In active mode, the V_{DDC} detector in EVR detects brownout when the core supply voltage V_{DDC} dips below

- the threshold voltage V_{DDC_TH} (approximately 2.3 V) or
- the threshold voltage V_{DDP_TH} (approximately 2.9 V) if the detector is enabled or
- the threshold voltage V_{DDP_TH} (approximately 2.4 V) if the detector is disabled

System Control Unit

The brownout will cause the device to be reset. In power-down mode, the V_{DDC} is monitored by the POR in EVR and a reset is generated when V_{DDC} drops below 1.5 V.

Once the brownout reset takes place, the reset sequence is the same as the power-on reset sequence.

7.2.2 Module Reset Behavior

Table 7-1 lists the functions of the XC82x and the various reset types that affect these functions. The symbol "■" signifies that the particular function is reset to its default state.

Table 7-1 Effect of Reset on Modules/Functions

Module/ Function	Power-On Reset	Brown-Out Reset	Wake-up Reset	Soft Reset	Watchdog Reset
CPU Core	■	■	■	■	■
SCU	■	■	■ except indication bits	■ except indication bits	■ except indication bits
Peripherals	■	■	■	■	■
Debug System	■	■	■	■	■
Port Control	■	■	■	■	■
FW Startup Execution	Executes all INIT	Executes all INIT	Executes all INIT	Executes all INIT	Executes all INIT
On-Chip Static RAM	Affected, unreliable	Affected, unreliable	Not affected, reliable	Not affected, reliable	Not affected, reliable
Flash	■	■	■	■	■
EVR	■	■	■	■	■
Clock System	■	■	■	■	■

System Control Unit

7.2.3 Reset Control Register Description

RSTCON register consist of the indication bits of a wake-up event, WDT reset and soft reset. [Table 7-2](#) shows the reset value of RSTCON register after these events.

RSTCON

Reset Control Register

(F7_H)

Reset Value: 00_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
SWRQ		0			SOFTRS	WDTRST	WKRS
rwh		r			rwh	rwh	rwh

Field	Bits	Type	Description
WKRS	0	rwh	Wake-up Indication Bit 0 No Wake-up occurred. 1 Wake-up has occurred. This bit can only be set by hardware and clear by software.
WDTRST	1	rwh	Watchdog Timer Reset Indication Bit 0 No watchdog reset occurred. 1 Watchdog reset has occurred. This bit can only be set by hardware and clear by software.
SOFTRS	2	rwh	Soft Reset Indication Bit 0 No soft reset occurred. 1 Soft reset has occurred. This bit can only be set by hardware and clear by software.
SWRQ	7	rwh	Soft Reset Request 0 No action. 1 On set, soft reset is requested. This bit is automatically cleared by hardware. The SWRQ bit is a protected bit. When the Protection Scheme is activated, this bit cannot be written directly.
0	[6:3]	r	Reserved Returns 0 if read; should be written with 0.

Table 7-2 Reset Value of Register RSTCON

Reset Source	Reset Value
Power-down Wake-up Reset	0000 0001 _B
WDT Reset	0000 0010 _B
Soft Reset	0000 0100 _B
Power-On Reset/Brown-out Reset	0000 0000 _B

7.3 Clock System and Control

Figure 7-2 shows the block diagram of the clock system in XC82x. It consists of a 48 MHz oscillator and a clock control unit (CCU). The system clock f_{SYS} is generated by the 48 MHz internal oscillator. In addition, f_{SYS} is also the input clock to the Clock Control Unit (CCU).

The CCU generates all clock signals required within the microcontroller from the system clock. It consists of:

- Clock mode selection in active mode
- Centralized enable/disable circuit for clock control

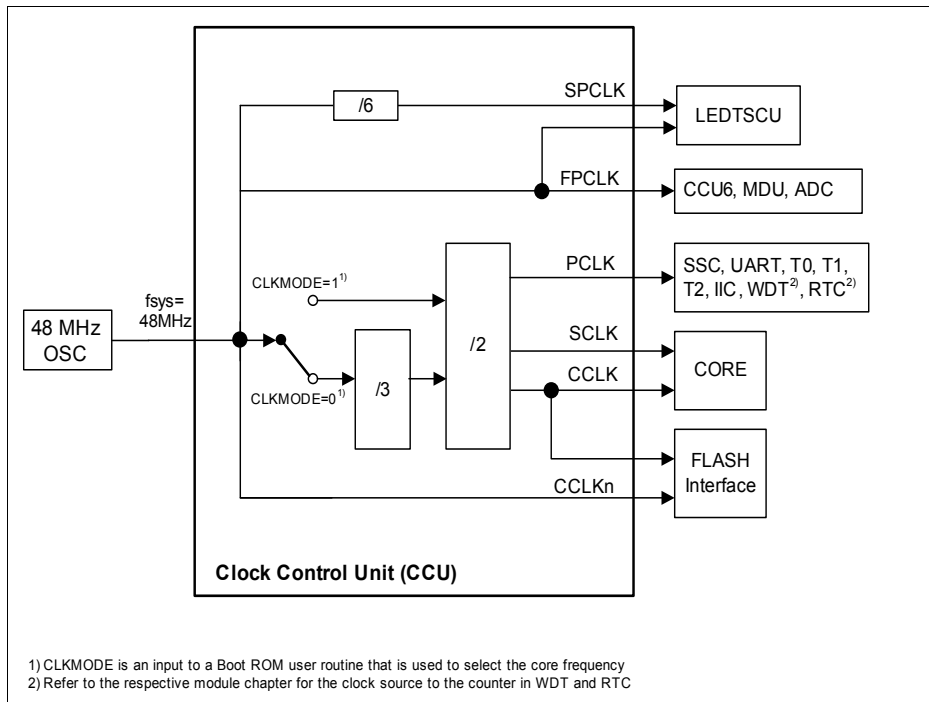


Figure 7-2 Clock System Block Diagram

In normal running mode, the typical frequencies of different modules are as follows:

- CPU clock (CCLK, SCLK) = 24 MHz
- Flash Interface clock (CCLKn) = 48 MHz
- Fast peripheral clock (FPCLK) = 48 MHz
- Slow peripheral clock (SPCLK) = 8 MHz
- Peripheral clock (PCLK) = 24 MHz (same as CPU clock)

System Control Unit

Peripherals that are running in FPCLK frequency are CCU6, MDU, ADC, LED and Touch-sense controller. Other peripherals are clocked by PCLK which is the same as CPU clock. In normal running mode, PCLK= CCLK.

In active mode, there are 2 clocking frequencies, 8 MHz and 24 MHz, for the CPU clock (CCLK, SCLK) and the peripherals clock (PCLK). A user routine called BR_CLKMODE_SETTING in the Boot ROM is used to switch between these 2 clock frequencies. If the CLKMODE input of the user routine is set to 0, 8 MHz frequency is selected. 24 MHz is selected when CLKMODE is set to 1. While changing the frequency of CPU clock and PCLK clock, it is recommended to disable all interrupts to prevent any access to flash that may results in an unsuccessful flash operation.

In idle mode, only the CPU clock CCLK is disabled. In power-down mode, CCLK, SCLK, FPCLK, SPCLK, CCLKn and PCLK are all disabled.

7.3.1 Oscillator Watchdog

There are 2 oscillator watchdogs in XC82x, namely, the 48 MHz oscillator watchdog (48 MHz OWD) and 75 KHz oscillator watchdog (75 KHz OWD). The 48 MHz OWD monitors the 48 MHz clock source. Only incoming frequencies that are below 40 MHz are detected. The 75 KHz OWD monitors the 75 KHz clock source. By setting bit OSC_CON.RCOWDRST, the detection for 48 MHz oscillator and 75 KHz oscillator can be restarted. The detection status output is only valid after 13 cycles of the 75 KHz frequency (approximately 180 us).

After reset, both the 48 MHz and 75 KHz OWD are default disabled. User need to check the status of OSC_CON.INTOSC_ST status flag before both OWDs can be enabled. When a 1 is detected in bit INTOSC_ST, the 75 KHz oscillator should be running in a stable trimmed frequency. Next, the OWD function could be enabled by setting bit RCOWDRST in OSC_CON register. User code is recommended to continue when 48MOSC2L and 75KOSC2L is both 0. User can also choose to continue with other operations while waiting for a stable 48MOSC2L and 75KOSC2L. The loss of oscillator clock NMI would be based on these 2 signals to trigger a NMI if it is enabled via bit NMICON.NMIOSCCLK.

7.3.2 Loss of Clock Detection and Recovery

Loss of clock happens when the 48 MHz oscillator watchdog detects a frequency that is less than 40 MHz during normal operation. In this case, an NMI interrupt will be generated if it is enabled by NMICON.NMIOSCCLK. Concurrently, the oscillator status flag, 48MOSC2L, is set to 1 and the system clock will be provided by a stable 75 KHz oscillator. Emergency routines can be executed with the XC82x clocked with this oscillator. In case of a malfunction 75 KHz oscillator, there will be no switching and the system clock source remains to be the 48 MHz oscillator. However, the 48 MHz oscillator watchdog will not be functioning. No monitoring of this master clock is possible.

System Control Unit

When the 75 KHz oscillator is detected to be malfunction, the 75 KHz oscillator watchdog status flag, 75KOSC2L, is set to 1. An NMI interrupt will be generated if it is enabled by NMICON.NMIOSCCLK. For both cases of oscillator failure, emergency routines can be executed using either 48 MHz or 75 KHz clock source.

The XC82x remains in this loss of clock state until the next power on reset or after a successful clock recovery has been performed. A clock recovery could be carried out by restarting the detection. Upon detecting a stable oscillator frequency of more than 40 MHz, 48MOSC2L will be set to 0 and the system clock will be switched automatically to the 48 MHz clock source.

Note: With 75 KHz as the system frequency in loss of clock state, read from flash is possible. However, Flash program and erase is not allowed.

Note: NMICON.NMIOSCCLK is used to enable the NMI interrupt for loss of clock failure in 48 MHz or 75 KHz oscillators.

System Control Unit

7.3.3 CCU Register Description

The registers of the clock control unit are reset to the default value after any type of reset. Register OSC_CON controls the 48 MHz oscillator watchdog and 75 KHz oscillator watchdog. .

OSC_CON

OSC Control Register

(F4_H)

Reset Value: 00_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	INTOSC_ST		0		75KOSC2L	48MOSC2L	RCOWDRST
r	rh		r		rh	rh	rwh

Field	Bits	Type	Description
RCOWDRST	0	rwh	48 MHz and 75 KHz Oscillators Watchdog Reset Setting this bit will restart the oscillator detection. This bit will be automatically reset to 0 and thus always be read back as 0. 0 No effect. 1 Restart the oscillator watchdog for 48 MHz and 75 KHz oscillation. 48MOSC2L and 75KOSC2L flag will be held in the last value until it is updated after 180 usec.
48MOSC2L	1	rh	48 MHz Oscillator Too Low Flag The Oscillator Watchdog monitors the f_{SYS} (same as f_{OSC}). System frequency could be generated from the internal 48 MHz oscillator or an external 48 MHz clock source. 0 f_{SYS} is above threshold. 1 f_{SYS} is below threshold.
75KOSC2L	2	rh	75 KHz Oscillator Too Low Flag The Oscillator Watchdog monitors the 75 KHz oscillation. 0 75 KHz oscillates above threshold. 1 75 KHz oscillates below threshold.
INTOSC_ST	6	rh	Internal Oscillator Stable Indication 0 48MHz and 75 KHz oscillators are not stable. 1 48MHz and 75 KHz oscillators are stable.

System Control Unit

Field	Bits	Type	Description
0	[5:3], 7	r	Reserved Returns 0 if read; should be written with 0.

Note: The reset value of OSC_CON register is 0000 0110_B. One clock after reset, bits 48MOSC2L and 75KOSC2L will be set to 0 if both oscillators are running, then the value 0000 0000_B will be observed.

7.4 Power Management

The power saving modes in the XC82x provide flexible power consumption through a combination of techniques, including:

- Stopping the CPU clock
- Stopping the clocks of individual system components
- Reducing clock speed of some peripheral components
- Power-down of the entire system with fast restart capability

After a reset, the active mode (normal operating mode) is selected by default (see [Figure 7-3](#)) and the system runs in the main system clock frequency. In active mode, the system clock could be running in 24 MHz or 8 MHz. From active mode, different power saving modes can be selected by software. They are:

- Idle mode
- Power-down mode 1
- Power down mode 2

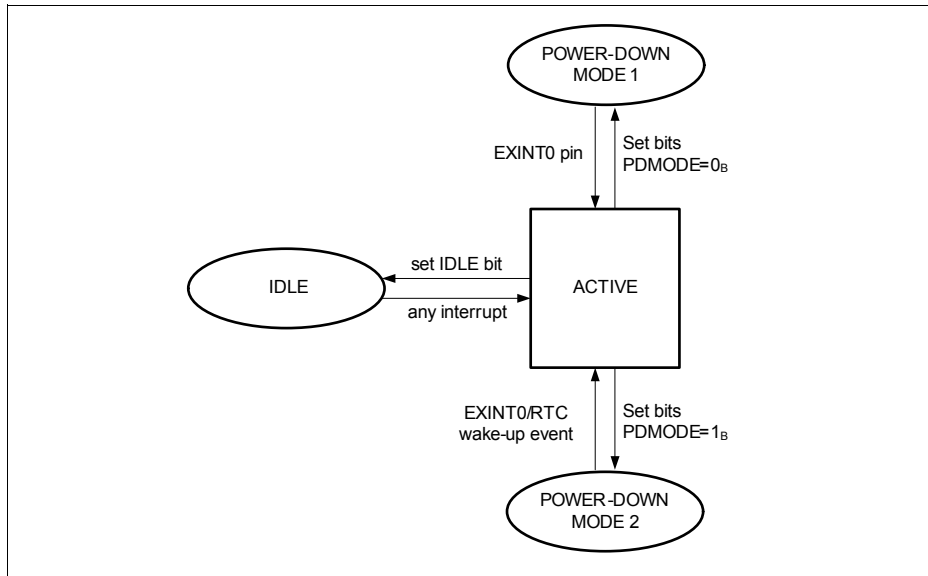


Figure 7-3 Transition between Power Saving Modes (without reset)

In XC82x, all functions must be operational in the active mode and idle mode when the normal V_{DDP} supply range of 3.0 V to 5.5 V is applied to the system. For reduced voltage condition in active mode and idle mode, required functions as needed by user will be available as long as the active current is kept under the limits. As for power down mode,

specified modules as described in [Section 7.4.1.2](#) must continue to function when the V_{DDP} is as low as 2.5 V but maybe performance could be reduced.

7.4.1 Functional Description

7.4.1.1 Idle Mode

The idle mode is used to reduce power consumption by stopping the core's clock.

In idle mode, the oscillator continues to run, but the core is stopped with its clock disabled. Peripherals whose input clocks are not disabled are still functional. The user should disable the Watchdog Timer (WDT) before the system enters the idle mode; otherwise, it will generate an internal reset when an overflow occurs and thus will disrupt the idle mode. The CPU status is preserved in its entirety; the stack pointer, program counter, program status word, accumulator, and all other registers maintain their data during idle mode. The port pins hold the logical state they had at the time the idle mode was activated.

Software requests idle mode by setting the bit PCON.IDLE to 1.

The idle mode can be terminated and the system will returned to active mode by activating any enabled interrupt. The CPU operation is resumed and the interrupt will be serviced. Upon RETI instruction, the core will return to execute the next instruction after the instruction that sets the IDLE bit to 1.

7.4.1.2 Power Down Mode

In order to achieve different levels of power saving, the XC82x has two types of power down modes, Power Down mode 1 and 2. Generally, the 48 MHz oscillator and the Flash memory are put into power down state in all the modes. In addition, the main voltage regulator is switched off, with only low power voltage regulator still operating in these power down modes. Therefore, most of the functions of the microcontroller are stopped while the contents of the Flash, on-chip RAM, XRAM, and the SFRs are maintained.

Table 7-3 shows the behaviour of several modules during power down mode. In power down mode 2, RTC is running in the periodic wake-up mode. It allows the device to exit power down mode at a specific period of time. The function is useful in low power application. Modules running in power down mode remain functioning at the reduced voltage condition where the main power supply could be as low as 2.5 V. However, it may shows a reduced performance. The rest of the modules are powered down in all power down modes. **Table 7-4** shows the available wake-up source for each power down mode. One of the available source is to receives an external wake-up signal via EXINT0 pin by setting bit EWS to 1. The edge that trigger the wake-up signal will depends on bit EXINT0 in EXICON0 register.

System Control Unit

Note: $EXICON0.EXINT0 = 11_B$ cannot be used to wake-up from power down mode.

Table 7-3 Modules Behavior in Power Down Mode¹⁾

Modules	Power Down Mode 1	Power Down Mode 2 ²⁾
RTC	N	Y
75 KHz OSC	N	Y
75 KHz OWD	N	N

1) N indicates that the module is shut down and Y indicates that it can run (if enabled) in power down mode.

2) RTC Mode 3 is not supported in power down mode. The RTCCLK pin will be shut down once power down mode is entered.

Table 7-4 Available Wake-up Source for Power Down Mode

Modules	Power Down Mode 1	Power Down Mode 2
RTC wake-up	Not available	Available
Clock Failure	Not available	Not available
EXINT0¹⁾	Available	Available

1) EXINT0 wake-up source is selected using bit PMCON0.EWS

Power Down Mode 1

All peripheral blocks and CPU including the real-time clock that operates with the 75 KHz oscillator is stopped. 75 KHz oscillator watchdog is also stopped. With the clock being turned off, the system cannot be awakened by an interrupt or the Watchdog Timer. It will be awakened only when it receives an external wake-up signal via EXINT0 pin by setting bit EWS to 1. The wake-up source and wake-up type must be selected before the system enters the power-down mode.

The sequence to enter power down mode 1 is:

- Select power down mode 1 by setting bit PMCON0.PDMODE to 0_B .
- Power down all modules including the 48 MHz and 75 KHz oscillator by setting bit PMCON0.PD to 1.

Three NOP instructions must be inserted after the bit PMCON0.PD is set to 1. This ensures the first instruction (after two NOP instructions) is executed correctly after wake-up from power-down mode.

Power Down Mode 2

In this mode, all the modules except those that are described in [Table 7-3](#) are powered down. The real-time clock that operates with the 75 KHz oscillators is running in the periodic wake-up mode and the real-time count is maintained. However, no monitoring

System Control Unit

of the status of the 75 KHz oscillation is possible. This is because the 75 KHz oscillator watchdog is powered down. To exit power down mode 2, the real-time clock wake up event can be used. Besides this wake-up source, it can also be awakened when it receives an external wake-up signal via EXINT0 pin by setting bit PMCON0.EWS to 1.

The sequence to enter power down mode 2 is:

- Ensure that real-time clock is enabled by setting bit RTCON.RTCC set to 1.
- Disable the WDT module by setting bit WDTCON.WDTEN to 0.
- Select power down mode 2 by setting PMCON0.PDMODE to 1_B.
- Enter power down mode by setting bit PMCON0.PD to 1.

Three NOP instructions must be inserted after the bit PMCON0.PD is set to 1. This ensures the first instruction (after two NOP instructions) is executed correctly after wake-up from power-down mode.

For all types of power down mode, the port pins hold the logical state they had when the power down mode was activated. For digital ports, the input and output driver of all port pins that are not used as wake-up source are disabled once the chip enters power down mode. This can reduce the leakage current in the power down mode.

Exiting Power Down Mode

Power down mode can be exited in various ways:

- The EXINT0 pin detects a edge based on the setting in bit EXICON0.EXINT0
- The wake-up event request from the real-time clock
- The RTC clock source failure

Note: EXICON0.EXINT0 = 11_B cannot be used to wake-up from power down mode

The EXINT0 wake-up source is enabled by PMCON0.EWS. Bit MODPSEL1.EXINT0IS can be used for the EXINT0 input pin selection. The wake-up with reset or without reset is selected by bit PMCON0.WKSEL.

If bit WKSEL was set to 1 before entering power-down mode, the system will execute a reset sequence similar to the power-on reset sequence. Therefore, all port pins are put into their reset state and will remain in this state until they are affected by program execution.

If bit WKSEL was cleared to 0 before entering power-down mode, a fast wake-up sequence is used. The port pins continue to hold their state which was valid during power-down mode until they are affected by program execution.

The wake-up from power-down without reset using EXINT0 wake-up source undergoes the following procedure:

1. In power-down mode, EXINT0 pin must be held at the inactive level.
2. Power-down mode is exited when EXINT0 pin goes active for at least 100 ns.
3. The main voltage regulator is switched on and takes approximately 150 µs to become stable.

System Control Unit

4. The on-chip oscillator is started. Typically, the on-chip oscillator takes approximately 10 μ s to stabilize.
5. Subsequently, the FLASH will enter ready-to-read mode. This does not require the typical 160 μ s as is the case for the normal reset. The timing for this part can be ignored.
6. The CPU operation is resumed. The core will return to execute the next instruction after the instruction which sets the PD bit.

Note: For this case, no interrupt will be generated by the EXINT0 wake-up source even if EXINT0 is enabled before entering power-down mode. It is the same for the rest of the wake-up sources. An interrupt will be generated only if EXINT0 fulfils the interrupt generation conditions after CPU resumes operation.

As for exiting power down mode to active mode in reduced voltage condition, the active current must be below the limits as specified in the Data Sheet. A brownout reset may occurred immediately after wakeup if the condition is not met.

After wake-up from power down mode without reset, the status flag of the wake-up event can be used to indicate the source of wake-up. In addition, the oscillator watchdog needs to be re-enabled if necessary by setting bit RCOWDRST in OSC_CON register. But before the oscillator watchdog can be enabled, it is required to ensure that the 48 MHz and 75 KHz oscillators are stable by checking the status of OSC_CON.INTOSC_ST status flag. See [Section 7.3.1](#) for more detail descriptions.

In power down mode 1, real-time clock is stopped and user is required to enabled it once the chip exit power down mode.

7.4.1.3 Peripheral Clock Management

The amount of reduction in power consumption that can be achieved by this feature depends on the number of peripherals running. Peripherals that are not required for a particular functionality can be disabled by gating off the clock inputs. For example, in idle mode, if all timers are stopped, and ADC, CCU6, MDU and the serial interfaces are not running, maximum power reduction can be achieved. However, the user must take care when determining which peripherals should continue running and which must be stopped during active and idle modes.

The ADC, SSC, CCU6, MDU, LEDTSCU, IIC, Timer 2 can be disabled (clock is gated off) by setting the corresponding bit in the PMCON1 register. Furthermore, the analog part of the ADC module may be disabled by resetting the GLOBCTR.ANON bit. This feature causes the generation of the ADC analog clock to be stopped and allows a reduction in power consumption when no conversion is needed.

PMCON1

Peripheral Management Control Register 1(EF_H)

Reset Value: DF_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	0	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	r	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ADC_DIS	0	rw	ADC Disable Request. Active high. 0 ADC is in normal operation. 1 Request to disable the ADC. (default)
SSC_DIS	1	rw	SSC Disable Request. Active high. 0 SSC is in normal operation. 1 Request to disable the SSC. (default)
CCU_DIS	2	rw	CCU Disable Request. Active high. 0 CCU is in normal operation. 1 Request to disable the CCU. (default)
T2_DIS	3	rw	T2 Disable Request. Active high. 0 T2 is in normal operation. 1 Request to disable the T2. (default)
MDU_DIS	4	rw	MDU Disable Request. Active high. 0 MDU is in normal operation. 1 Request to disable the MDU. (default)

System Control Unit

Field	Bits	Type	Description
LTS_DIS	6	rw	LEDTSCU Disable Request. Active high. 0 LEDTSCU is in normal operation. 1 Request to disable the LEDTSCU. (default)
IIC_DIS	7	rw	IIC Disable Request. Active high. 0 IIC is in normal operation. 1 Request to disable the IIC. (default)
0	5	r	Reserved Returns 0 if read; should be written with 0.

7.4.2 Power Management Register Description

PMCON0

Power Mode Control Register 0 (F3_H)

Reset Value: 01_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
	0		WKSEL	0	PDMODE	PD	EWS
	r		rw	r	rw	rwh	rw

Field	Bits	Type	Description
EWS	0	rw	External interrupt 0 Wake-up Source Selected 0 External interrupt 0 wake-up is not selected. 1 External interrupt 0 wake-up is selected.
PD	1	rwh	Power Down Mode Enable. Active High. Setting this bit will cause the chip to go into a power down mode as indicated by bit PDMODE. Reset by wake-up circuit. The PD bit is a protected bit. When the Protection Scheme is activated, this bit cannot be written directly. For more information on Protection Scheme, see Section 3.4.4 .
PDMODE	2	rw	Power Down Mode Select 0 Power down mode 1 is selected. 1 Power down mode 2 is selected.
WKSEL	4	rw	Wake-up Reset Select Bit 0 Wake-up without reset. 1 Wake-up with reset.
0	3, [7:5]	r	Reserved Returns 0 if read; should be written with 0.

System Control Unit
PCON [Note: This register is located within XC800 core]
Power Control Register [Not bitaddressable](87_H)
Reset Value: 00_H
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
SMOD	0			GF1	GF0	0	IDLE
rw	r			rw	rw	r	rw

Field	Bits	Type	Description
IDLE	0	rw	Idle Mode Enable 0 Do not enter Idle Mode 1 Enter Idle Mode

7.5 SCU Register Mapping

The system control SFRs are used to control the overall system functionalities, such as interrupts, variable baud rate generation, clock management, bit protection scheme and oscillator. The SFRs are located in the standard memory area (RMAP = 0) and are organized into 8 pages. The SCU_PAGE register is located at F1_H. It contains the page value and page control information.

SCU_PAGE

Page Register for SCU

(F1_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
OP		STNR		0	PAGE		
w		w		r	rwh		

Field	Bits	Type	Description
PAGE	[3:0]	rwh	Page Bits When written, the value indicates the new page address. When read, the value indicates the currently active page = addr [y:x+1]
STNR	[5:4]	w	Storage Number This number indicates which storage bit field is the target of the operation defined by bit OP. If OP = 10 _B , the contents of PAGE are saved in STx before being overwritten with the new value. If OP = 11 _B , the contents of PAGE are overwritten by the contents of STx. The value written to the bit positions of PAGE is ignored. 00 ST0 is selected. 01 ST1 is selected. 10 ST2 is selected. 11 ST3 is selected.

System Control Unit

Field	Bits	Type	Description
OP	[7:6]	w	Operation 0X Manual page mode. The value of STNR is ignored and PAGE is directly written. 10 New page programming with automatic page saving. The value written to the bit positions of PAGE is stored. In parallel, the former contents of PAGE are saved in the storage bit field STx indicated by STNR. 11 Automatic restore page action. The value written to the bit positions PAGE is ignored and instead, PAGE is overwritten by the contents of the storage bit field STx indicated by STNR.

The addresses of the system control SFRs are listed in [Table 7-5](#). Not listed in the tables is the register SYSCON0, which can be accessed in both standard (non-mapped) and mapped address of 8F_H.

Table 7-5 SFR Address List for SCU Pages 0 - 7

Address	Page 0	Page 1	Page 2	Page 3
F2 _H	IRCON0	PASSWD		XADDRH
F3 _H	IRCON1	PMCON0		MODPISEL
F4 _H	EXICON1	OSC_CON		MODPISEL1
F5 _H	IRCON2	ID		MODPISEL2
F6 _H	IRCON3	WDTCON		MODSUSP
F7 _H	NMISR	RSTCON		MODIEN
EE _H	NMICON	SDCON		MODPISEL3
EF _H	EXICON0	PMCON1		
Address	Page 4	Page 5	Page 6	Page 7
F2 _H		BCON		
F3 _H	WDTREL	BGL		
F4 _H	WDTWINB	BGH		
F5 _H	WDTL	LINST		
F6 _H	WDTH	FEAL		
F7 _H		FEAH		

8 Watchdog Timer

8.1 Overview

The Watchdog Timer (WDT) provides a highly reliable and secure way to detect and recover from software or hardware failures. The WDT is reset at a regular interval that is predefined by the user. The CPU must service the WDT within this interval to prevent the WDT from causing an XC82x system reset. Hence, routine service of the WDT confirms that the system is functioning properly. This ensures that an accidental malfunction of the XC82x will be aborted in a user-specified time period.

The WDT is by default disabled.

In debug mode, the WDT is default suspended and stops counting (its debug suspend bit is default set i.e., MODSUSP.WDTSUSP = 1. Therefore during debugging, there is no need to refresh the WDT.

Features

- 16-bit Watchdog Timer
- Programmable reload value for upper 8 bits of timer
- Programmable window boundary
- Input frequency from a secondary clock source of 75 KHz internal oscillator

Watchdog Timer

8.2 System Information

This section consist of the system information required to use the WDT.

8.2.1 Reset effects

The Watchdog Timer maintains a counter which must be refreshed or cleared periodically. Otherwise, the counter will overflow and the watchdog reset will be asserted. The occurrence of a WDT reset is indicated by the bit WDTRST in RSTCON register. The bit field of

The bit field PAGE of SCU_PAGE register must be programmed before accessing the RSTCON register.

RSTCON

Reset Control Register

(F7_H)

Reset Value: 00_H¹⁾

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
SWRQ	0				SOFTRS	WDTRST	WKRS
rwh	r				rwh	rwh	rwh

1) The reset value for watchdog timer reset is 02_H.

Field	Bits	Type	Description
WDTRST	1	rwh	Watchdog Timer Reset Indication Bit 0 No watchdog reset occurred. 1 Watchdog reset has occurred. This bit can only be set by hardware and clear by software.
0	[6:3]	r	Reserved Returns 0 if read; should be written with 0.

8.2.2 Clocking Configuration

The WDT runs on the 75 KHz Oscillator.

8.2.3 Interrupt Events and Assignment

Table 8-1 shows the non-maskable interrupt node assignment of the WDT interrupt source.

Watchdog Timer

Table 8-1 WDT Events' Non-maskable Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
WDT Overflow	NMICON.NMIWDT	NMISR.FNMIWDT	73 _H

8.2.4 Module Suspend Control

The timer in WDT is by default suspended on entering debug mode. The WDT can be allowed to run in debug mode by clearing the bit WDTSUSP in SFR MODSUSP to 0. Refer to [Chapter 10.2.4](#) for the definition of register MODSUSP.

8.3 Functional Description

The Watchdog Timer (WDT) is a 16-bit timer, which is incremented by a count rate of 75 KHz clock. This 16-bit timer is realized as two concatenated 8-bit timers. The upper 8 bits of the WDT can be preset to a user-programmable value via a watchdog service access in order to vary the watchdog expire time. The lower 8 bits are reset on each service access. **Figure 8-1** shows the block diagram of the WDT unit.

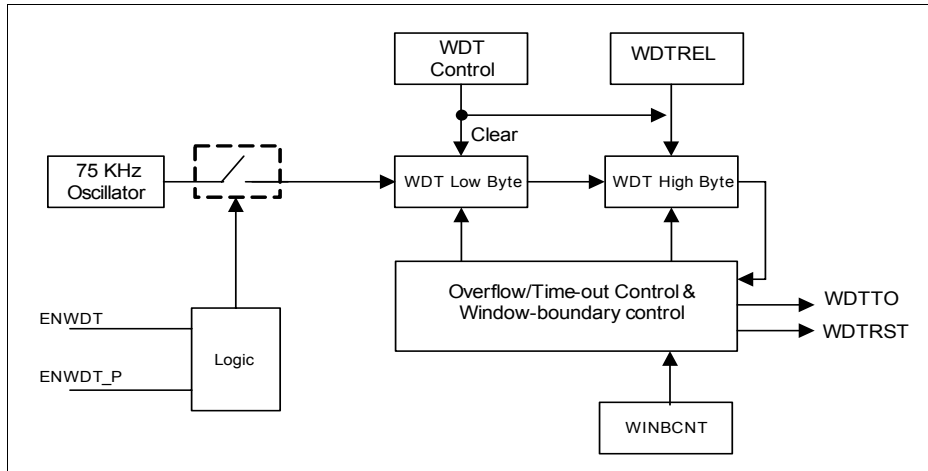


Figure 8-1 WDT Block Diagram

If the WDT is enabled by setting WDTEN to 1, the timer is set to a user-defined start value and begins counting up. It must be serviced before the counter overflows. Servicing is performed through refresh operation (setting bit WDTRS to 1). This reloads the timer with the start value, and normal operation continues.

If the WDT is not serviced before the timer overflows, a system malfunction is assumed and normal mode is terminated. A WDT NMI request (FNMIWDT) is then asserted and prewarning is entered. The prewarning lasts for 30_H count. During the prewarning period, refreshing of the WDT is ignored and the WDT cannot be disabled. A reset (WDTRST) of the XC82x is imminent and can no longer be avoided. The occurrence of a WDT reset is indicated by the bit WDTRST in RSTCON register. If refresh happens at the same time an overflow occurs, WDT will not go into prewarning period.

The WDT must be serviced periodically so that its count value will not overflow. Servicing the WDT clears the low byte and reloads the high byte with the preset value in bit field WDTREL. Servicing the WDT also clears the bit WDTRS.

The WDT has a “programmable window boundary”, which disallows any refresh during the WDT’s count-up. A refresh during this window-boundary constitutes an invalid access to the WDT and causes the WDT to activate WDTRST, although no NMI request

Watchdog Timer

is generated in this instance. The window boundary is from 0000_H to the value obtained from the concatenation of WDTWINB and 00_H . This feature can be enabled by WINBEN. After being serviced, the Watchdog Timer continues counting up from the value ($<WDTREL> \times 2^8$). The time period for an overflow of the Watchdog Timer is programmable by the reload value, WDTREL. It is the high byte of WDT and can be programmed in register WDTREL.

The period P_{WDT} between servicing the WDT and the next overflow can be determined by the following formula:

$$P_{WDT} = \frac{(2^{16} - WDTREL \times 2^8)}{f_{PCLK}} \quad (8.1)$$

If the Window-Boundary Refresh feature of the WDT is enabled, the period P_{WDT} between servicing the WDT and the next overflow is shortened if WDTWINB is greater than WDTREL. See also [Figure 8-2](#). This period can be calculated by the same formula by replacing WDTREL with WDTWINB. In order for this feature to be useful, WDTWINB cannot be smaller than WDTREL.

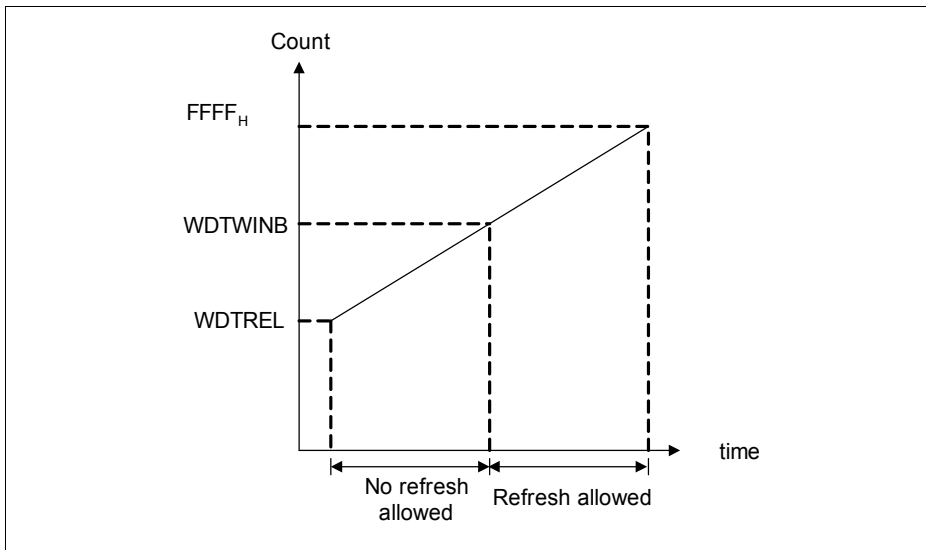


Figure 8-2 WDT Timing Diagram

Table 8-2 lists the possible ranges for the watchdog time which can be achieved using a certain module clock. Some numbers are rounded to 3 significant digits.

Watchdog Timer

Table 8-2 Watchdog Time Ranges

Reload Value in WDTREL	75 KHz Input frequency
FF _H	3.4 ms
7F _H	440 ms
00 _H	874 ms

Note: For safety reasons, the user is advised to rewrite WDTCN each time before the WDT is serviced.

Note: The Watchdog Timer can be suspended when OCDS enters Monitor Mode and has the Debug-Suspend signal activated, provided the respective suspend bit, WDTSUSP in SFR MODSUSP, are set. See Module Suspend Control section.

Watchdog Timer

8.4 Registers Description

Five SFRs control the operations of the WDT. They can be accessed from the mapped SFR area.

Table 8-3 lists the addresses of these SFRs.

Table 8-3 Register Map

Address	Page	Register
F6 _H	1	WDTCN
F3 _H	4	WDTREL
F4 _H	4	WDTWINB
F5 _H	4	WDTL
F6 _H	4	WDTH

8.4.1 Watchdog Timer Registers

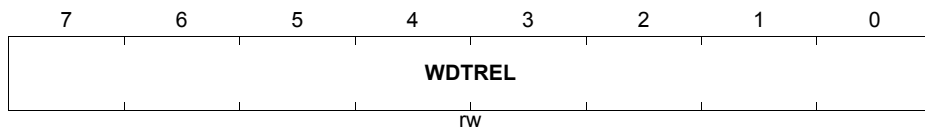
The Watchdog Timer Current Count Value is contained in the Watchdog Timer Register WDTH and WDTL, which are non-bitaddressable read-only register. The operation of the WDT is controlled by its bitaddressable WDT Control Register WDTCN. This register also selects the input clock prescaling factor. The register WDTREL specifies the reload value for the high byte of the timer. WDTWINB specifies Watchdog Window-Boundary count value.

WDTREL

Watchdog Timer Reload Register (F3_H)

Reset Value: 00_H

RMAP: 0, PAGE: 4



Field	Bits	Type	Description
WDTREL	[7:0]	rw	Watchdog Timer Reload Value (for the high byte of WDT)

Watchdog Timer

WDTCON

Watchdog Timer Control Register (F6_H)

Reset Value: 00_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	WINBEN	WDTPR	0	WDTEN	WDTRS	0	
r	rw	rh	r	rw	rwh	r	

Field	Bits	Type	Description
WDTRS	1	rwh	WDT Refresh Start Active high. Set to start refresh operation on the watchdog timer. Cleared automatically by hardware after it is set by software.
WDTEN	2	rw	WDT Enable 0 WDT is disabled 1 WDT is enabled WDTEN is a protected bit. If the Protection Scheme is activated then this bit cannot be written directly. See protection Scheme for more details.. <i>Note: Clearing WDTEN bit to 0 during Prewarning Mode (WDTPR = 1) has no effect.</i>
WDTPR	4	rh	Watchdog Prewarning Mode Flag 0 Normal mode (default after reset) 1 The Watchdog is operating in Prewarning Mode This bit is set to 1 when a Watchdog error is detected. The Watchdog Timer has issued an NMI trap and is in Prewarning Mode. A reset of the chip occurs after the prewarning period has expired.
WINBEN	5	rw	Watchdog Window-Boundary Enable 0 Watchdog Window-Boundary feature is disabled. (default) 1 Watchdog Window-Boundary feature is enabled.
0	0, 3, [7:6]	r	Reserved Returns 0 if read; should be written with 0.

Watchdog Timer

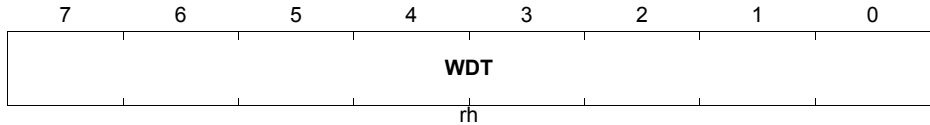
WDTL

Watchdog Timer, Low Byte

(F5_H)

Reset Value: 00_H

RMAP: 0, PAGE: 4



Field	Bits	Type	Description
WDT	[7:0]	rh	Watchdog Timer Current Value

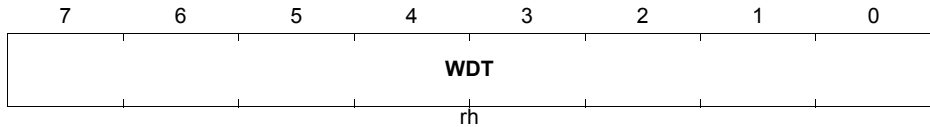
WDTH

Watchdog Timer, High Byte

(F6_H)

Reset Value: 00_H

RMAP: 0, PAGE: 4



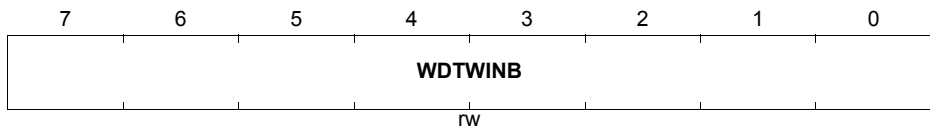
Field	Bits	Type	Description
WDT	[7:0]	rh	Watchdog Timer Current Value

WDTWINB

Watchdog Window-Boundary Count (F4_H)

Reset Value: 00_H

RMAP: 0, PAGE: 4



Watchdog Timer

Field	Bits	Type	Description
WDTWINB	[7:0]	rw	Watchdog Window-Boundary Count Value This value is programmable. Within this Window-Boundary range from 0000 _H to (WDTWINB, 00 _H), the WDT cannot do a Refresh, else it will cause a WDTRST to be asserted. WDTWINB is matched to WDTH.

9 Interrupt System

The XC800 Core supports one non-maskable interrupt (NMI) and 14 maskable interrupt requests. In addition to the standard interrupt functions supported by the core, e.g., configurable interrupt priority and interrupt masking, the XC82x interrupt system provides extended interrupt support capabilities such as the mapping of each interrupt vector to several interrupt sources to increase the number of interrupt sources supported, and additional status registers for detecting and identifying the interrupt source.

9.1 Interrupt Sources

The XC82x supports 14 interrupt vectors with four priority levels. Ten of these interrupt vectors are assigned to the on-chip peripherals: Timer 0, Timer 1, UART and SSC are each assigned one dedicated interrupt vector; while Timer 2, A/D Converter, LIN, LEDTSCU and the Capture/Compare Unit share six interrupt vectors. In addition, four interrupt vectors are assigned to the external interrupts, MDU, RTC and IIC. External interrupts 0 to 1 are each assigned one dedicated interrupt vector. External interrupt 2 is shared with MDU and IIC. RTC and External interrupt [6:3] share the same interrupt vector.

A non-maskable interrupt (NMI) with the highest priority is shared by the following:

- Watchdog Timer, warning before overflow
- 48 MHz and 75 KHz Oscillators, loss of oscillator clock
- Flash Timer, on operation complete e.g. erase.
- OCDS, on user IRAM event
- Flash ECC error
- V_{DDP} prewarning
- V_{DDC} prewarning

Figure 9-1, **Figure 9-2**, **Figure 9-3**, **Figure 9-4** and **Figure 9-5** give a general overview of the interrupt sources and nodes, and their corresponding control and status flags. **Figure 9-6** gives the corresponding overview for the NMI sources.

Interrupt System

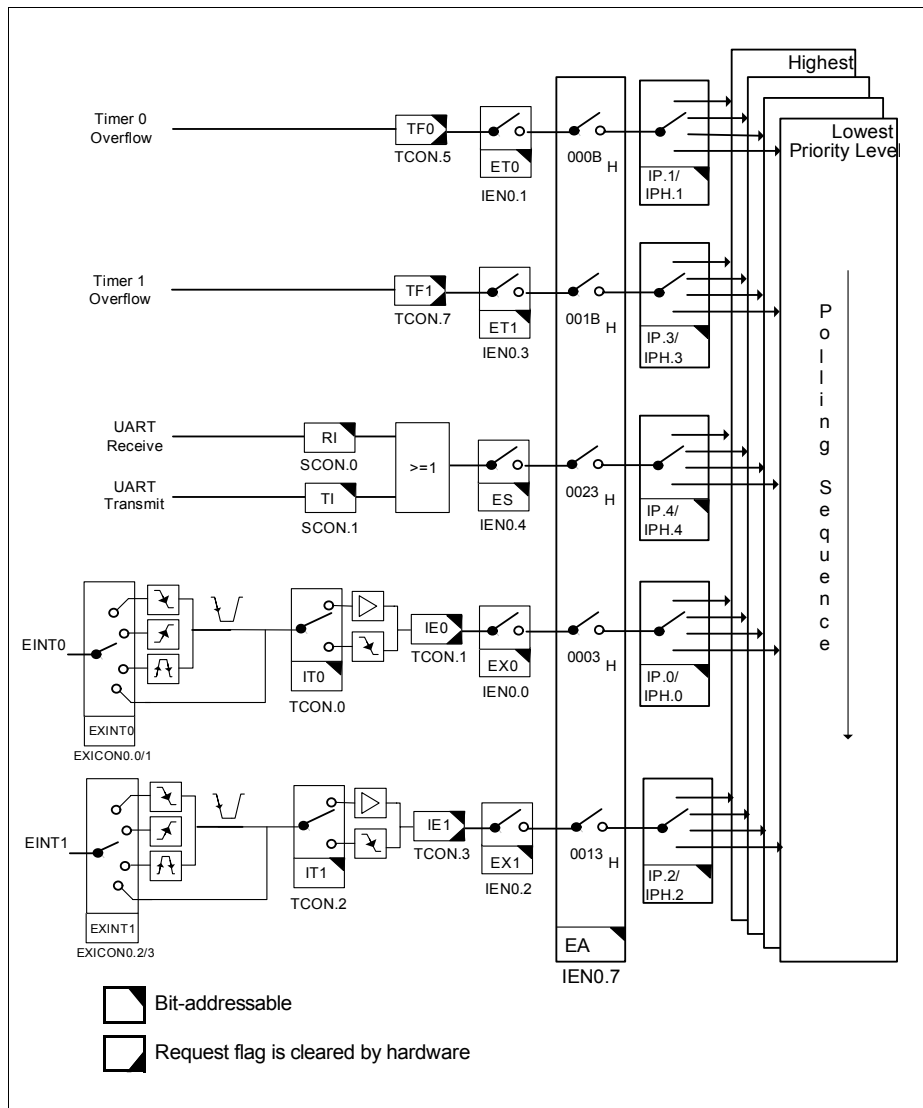


Figure 9-1 Interrupt Request Sources (Part 1)

Interrupt System

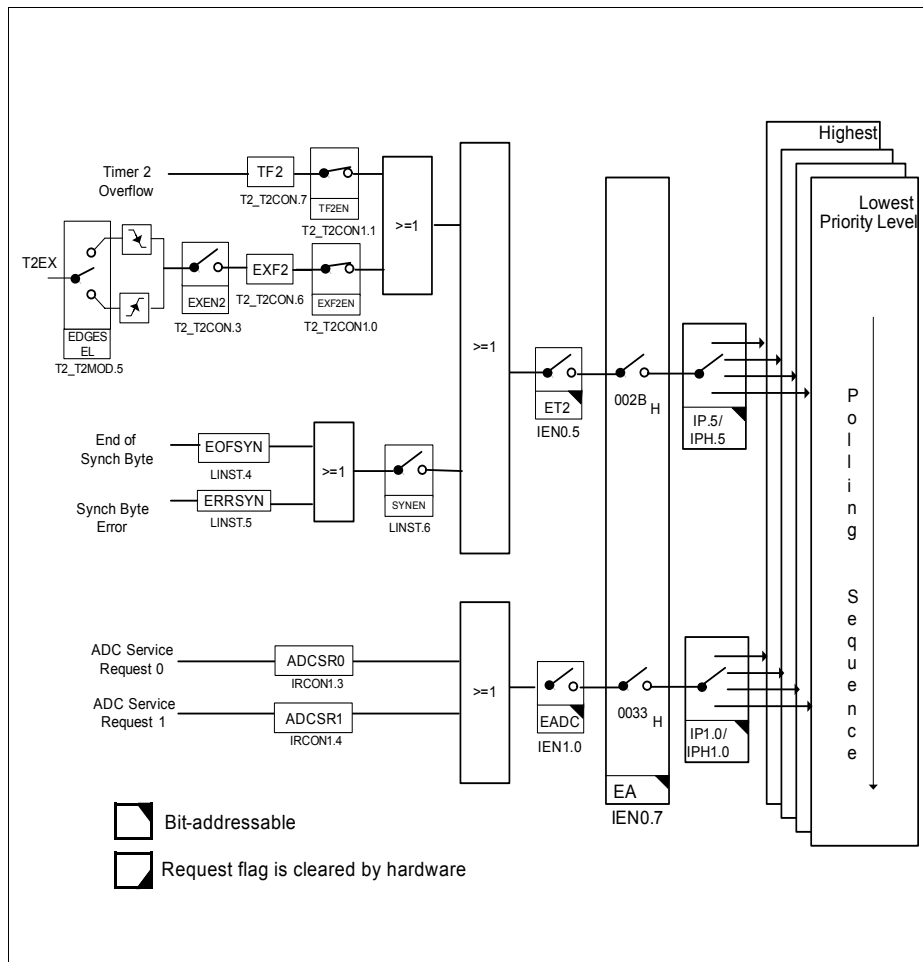


Figure 9-2 Interrupt Request Sources (Part 2)

Interrupt System

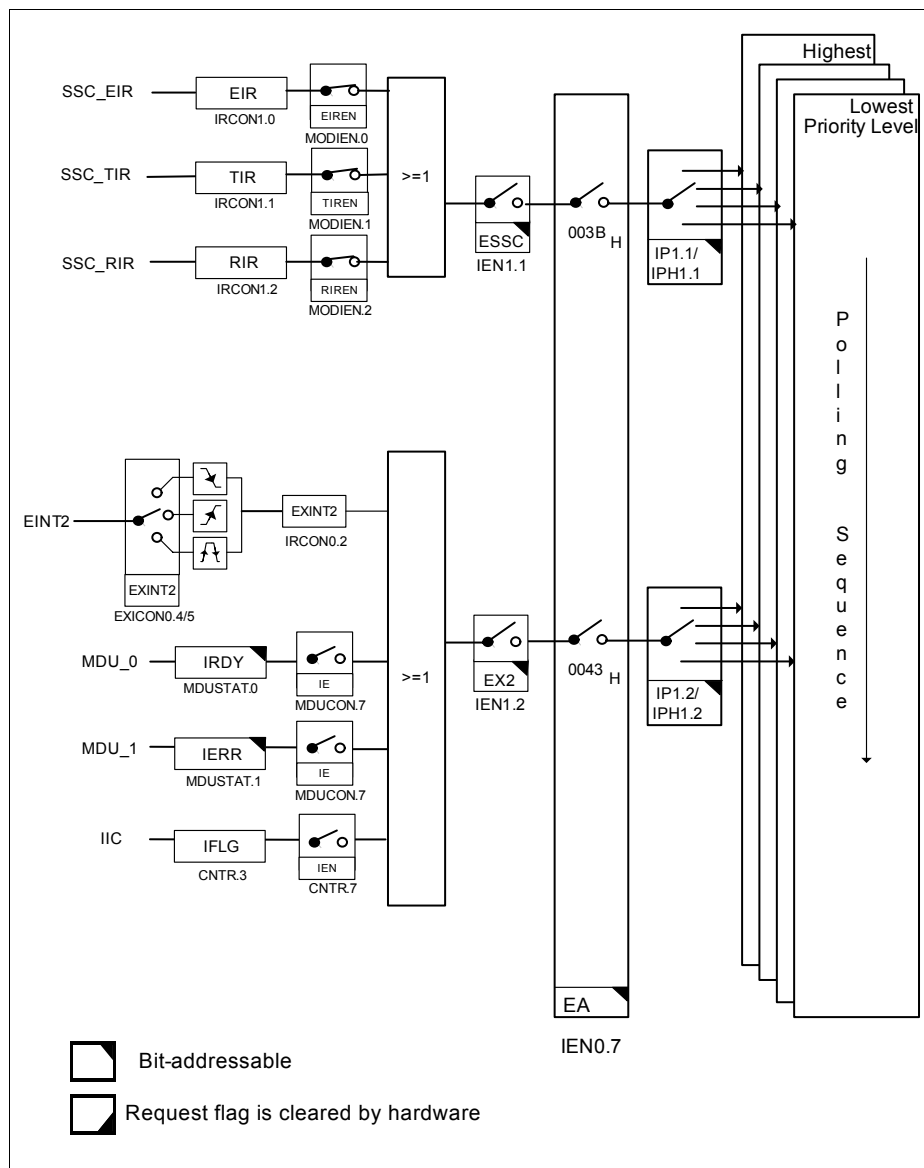


Figure 9-3 Interrupt Request Sources (Part 3)

Interrupt System

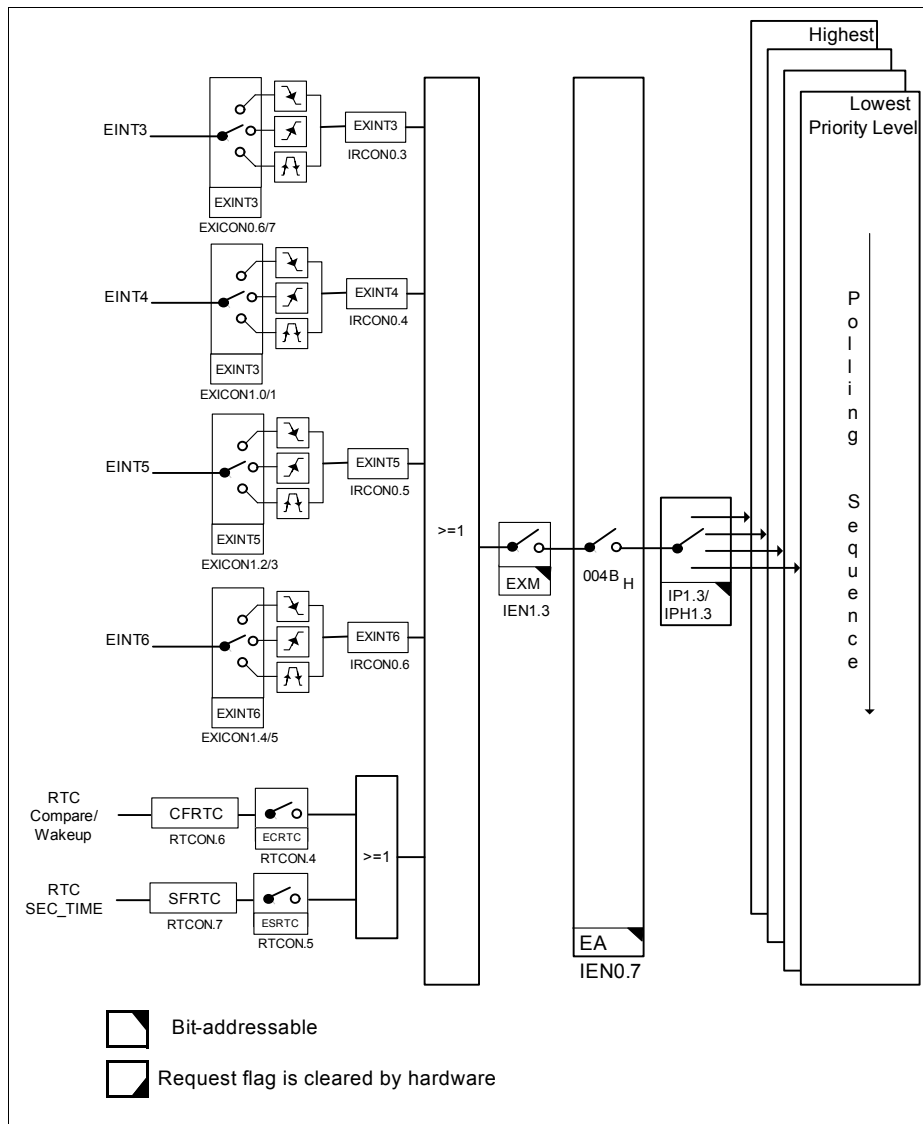


Figure 9-4 Interrupt Request Sources (Part 4)

Interrupt System

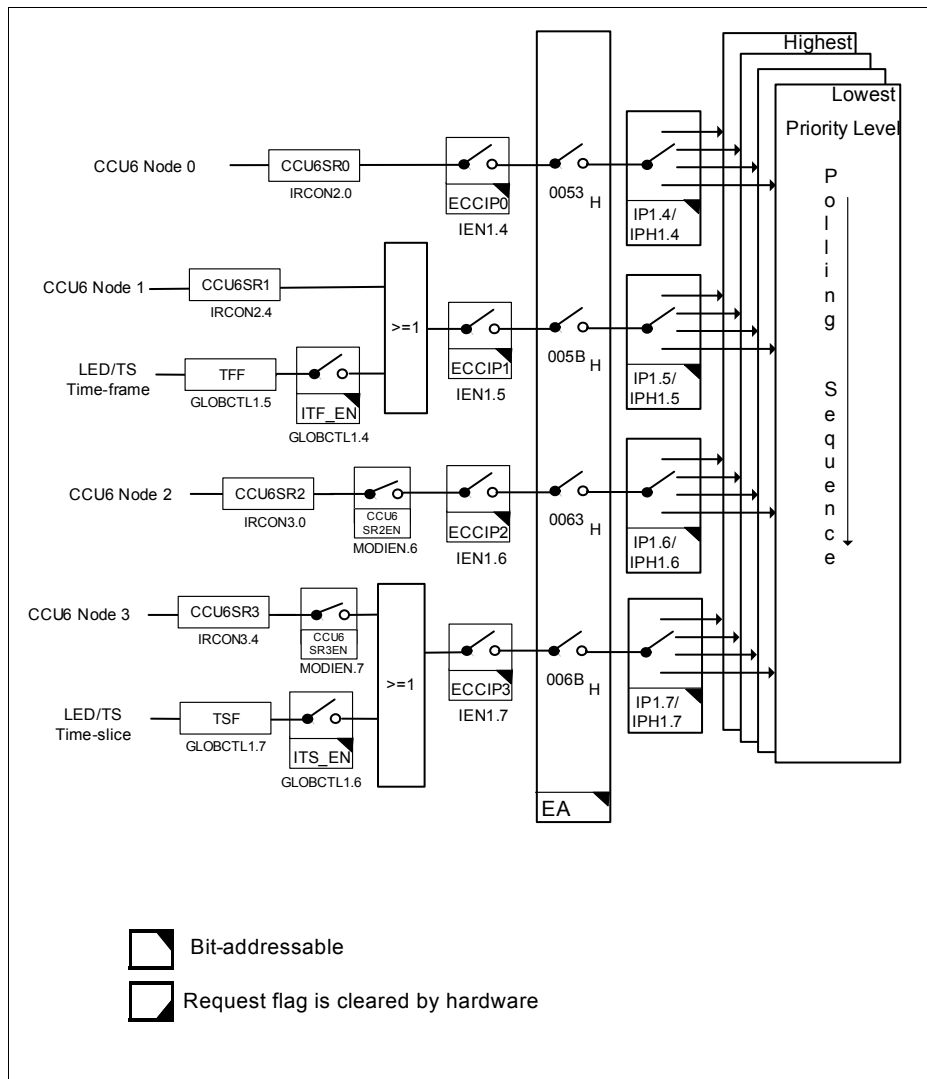


Figure 9-5 Interrupt Request Sources (Part 5)

9.1.1 Interrupt Source and Vector

Each interrupt event source has an associated interrupt vector address for the interrupt node it belongs to. This vector is accessed to service the corresponding interrupt node request. The interrupt service of each interrupt node can be individually enabled or disabled via an enable bit. The assignment of the XC82x interrupt sources to the interrupt vector address and the corresponding interrupt node enable bits are summarized in [Table 9-1](#).

Table 9-1 Interrupt Vector Address

Interrupt Node	Vector Address	Assignment for XC82x	Enable Bit	SFR
NMI	0073 _H	Watchdog Timer NMI	NMIWDT	NMICON
		48 MHz and 75 KHz clock NMI	NMIOSCCLK	
		Flash Operation Complete NMI	NMIFLASH	
		OCDS NMI	NMIOCDS	
		ECC Error NMI	NMIECC	
		VDDP Prewarning NMI	NMIVDDP	
		VDDC Prewarning NMI	NMIVDDC	
XINTR0	0003 _H	External Interrupt 0	EX0	IEN0
XINTR1	000B _H	Timer 0	ET0	
XINTR2	0013 _H	External Interrupt 1	EX1	
XINTR3	001B _H	Timer 1	ET1	
XINTR4	0023 _H	UART	ES	
XINTR5	002B _H	Timer2	ET2	
		LIN		

Table 9-1 Interrupt Vector Address (cont'd)

Interrupt Node	Vector Address	Assignment for XC82x	Enable Bit	SFR
XINTR6	0033 _H	ADC	EADC	IEN1
XINTR7	003B _H	SSC	ESSC	
XINTR8	0043 _H	External Interrupt 2	EX2	
		MDU		
		IIC		
XINTR9	004B _H	External Interrupt 3	EXM	
		External Interrupt 4		
		External Interrupt 5		
		External Interrupt 6		
		RTC Interrupt		
XINTR10	0053 _H	CCU6 SR0	ECCIP0	
XINTR11	005B _H	CCU6 SR1	ECCIP1	
		LEDTSCU Time Frame		
XINTR12	0063 _H	CCU6 SR2	ECCIP2	
XINTR13	006B _H	CCU6 SR3	ECCIP3	
		LEDTSCU Time Slice		

9.1.2 Interrupt Source and Priority

An interrupt that is currently being serviced can only be interrupted by a higher-priority interrupt, but not by another interrupt of the same or lower priority. Hence, an interrupt of the highest priority cannot be interrupted by any other interrupt request.

If two or more requests of different priority levels are received simultaneously, the request with the highest priority is serviced first. If requests of the same priority are received simultaneously, an internal polling sequence determines which request is serviced first. Thus, within each priority level, there is a second priority structure determined by the polling sequence as shown in [Table 9-2](#).

Table 9-2 Priority Structure within Interrupt Level

Source	Level
Non-maskable Interrupt (NMI)	(highest)
External Interrupt 0	1
Timer 0	2

Table 9-2 Priority Structure within Interrupt Level (cont'd)

Source	Level
External Interrupt 1	3
Timer 1	4
UART	5
Timer 2 / LIN	6
A/D Converter and ORC	7
SSC	8
External Interrupt 2 / MDU / IIC	9
EXINT[6:3] / RTC	10
CCU6 Interrupt Node Pointer 0	11
CCU6 Interrupt Node Pointer 1 / LED and Touch-sense	12
CCU6 Interrupt Node Pointer 2	13
CCU6 Interrupt Node Pointer 3 / LED and Touch-sense	14

9.2 Interrupt Structure

An interrupt event source may be generated from the on-chip peripherals or from external. Detection of interrupt events is controlled by the respective on-chip peripherals. Interrupt status flags are available for determining which interrupt event has occurred, especially useful for an interrupt node which is shared by several event sources. Each interrupt node (except NMI) has a global enable/disable bit. In most cases, additional enable bits are provided for enabling/disabling particular interrupt events (provided for NMI events). No interrupt will be requested for any occurred event that has its interrupt enable bit disabled.

The XC82x has, in general, two interrupt structures distinguished mainly by the manner in which the pending interrupt request (one per interrupt vector/node going directly to the core) is generated (due to the events) and cleared.

Common among these two interrupt structures is the interrupt masking bit, EA, which is used to globally enable or disable all interrupt requests (except NMI) to the core. Resetting bit EA to 0 only masks the pending interrupt requests from the core, but does not block the capture of incoming interrupt requests.

Note: The NMI node is similar to the other interrupt nodes, except for the exclusion of EA bit. Effectively, NMI node is non-maskable.

9.2.1 Interrupt Structure 1

For interrupt structure 1 (see [Figure 9-7](#)), the interrupt event will set the interrupt status flag which doubles as a pending interrupt request to the core. An active pending interrupt request will interrupt the core only if its corresponding interrupt node is enabled. Once an interrupt node is serviced (interrupt acknowledged), its pending interrupt request (represented by the interrupt status flag) may be automatically cleared by hardware (the core).

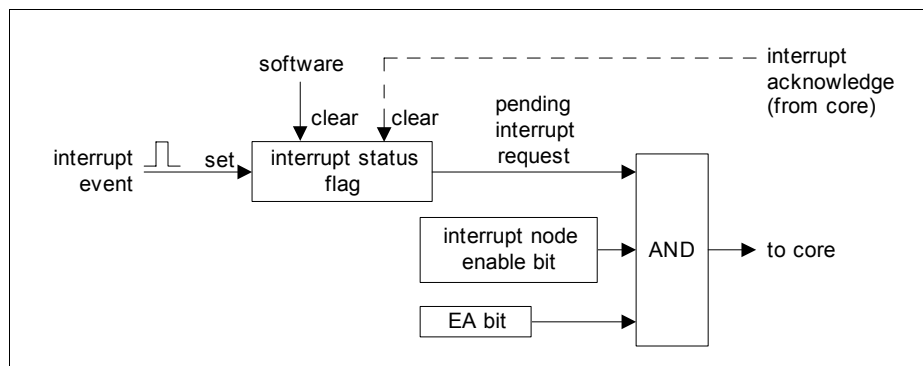


Figure 9-7 Interrupt Structure 1

For the XC82x, interrupt sources Timer 0, Timer 1, external interrupt 0 and external interrupt 1 (each have a dedicated interrupt node) will have their respective interrupt status flags TF0, TF1, IE0 and IE1 in register TCON cleared by the core once their corresponding pending interrupt request is serviced. In the case that an interrupt node is disabled (e.g. software polling is used), its interrupt status flag must be cleared by software since the core will not be interrupted (and therefore the interrupt acknowledge is not generated). For the UART which has its dedicated interrupt node, interrupt status flags RI and TI in register SCON will not be cleared by the core even when its pending interrupt request is serviced. The UART interrupt status flags (and hence the pending interrupt request) can only be cleared by software.

9.2.2 Interrupt Structure 2

This structure applies to the Timer 2, LIN, external interrupts 2 to 6, ADC, SSC, IIC, RTC, LEDTSCU, CCU6 and MDU interrupt sources. For interrupt structure 2 (see [Figure 9-8](#)), the interrupt status flag does not directly drive the pending interrupt request.

Interrupt System

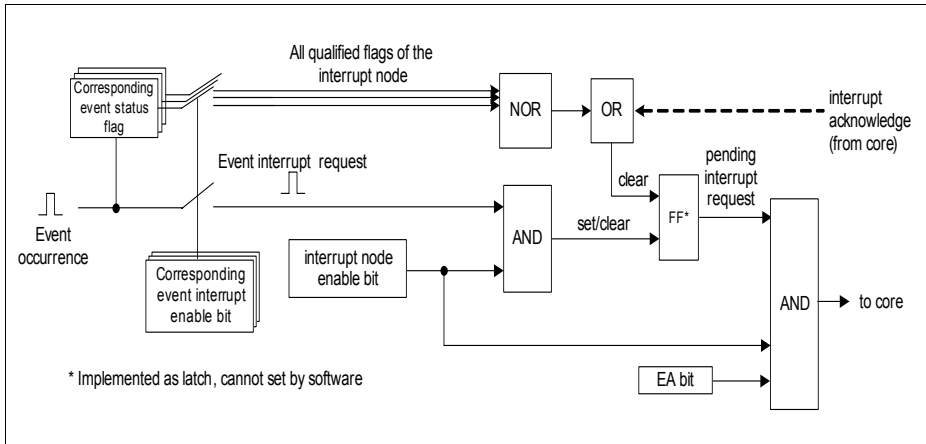


Figure 9-8 Interrupt Structure 2

An event generated by its corresponding interrupt source will set the status flag, and in parallel, if the event is enabled for interrupt, activate the pending interrupt request. Consider the case where interrupt event occurred while its interrupt node was disabled (assume global interrupt enable EA is set). When the interrupt node is enabled later, previously activated pending interrupt request will now cause an active interrupt request to the core. Note for the special case of NMI node, that it is essentially non-maskable.

An active pending interrupt request interrupts the core and is automatically cleared by hardware (the core) once the interrupt node is serviced (interrupt acknowledged); the status flag remains set and must be cleared by software. A pending interrupt request can also be cleared by software: only on clearing all interrupt-enabled status flags of the node will indirectly clear its pending interrupt request. Note that this is not exactly like interrupt structure 1 where the pending interrupt request is cleared directly by resetting the node's interrupt status flags.

In summary, the following lists in descending order the priority handling of pending interrupt request for interrupt nodes of structure 2: 1) CPU acknowledge of interrupt on vectoring to the service routine will clear the pending interrupt request, 2) Any event occurring and enabled for interrupt will set the pending interrupt request, 3) on clearing of all interrupt-enabled status flags will clear the pending interrupt request.

9.3 Interrupt Handling

The interrupt request signals are sampled at phase 2 in each machine cycle. The sampled requests are then polled during the following machine cycle. If one interrupt node request was active at phase 2 of the preceding cycle, the polling cycle will find it

Interrupt System

and the interrupt system will generate a LCALL to the node's service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of equal or higher priority is already in progress.
2. The current (polling) cycle is not in the final cycle of the instruction in progress.
3. The instruction in progress is RETI or any write access to registers IEN0/IEN1 or IP/IPH or IP1/IPH1.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress is completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any write access to registers IEN0/IEN1 or IP/IPH or IP1/IPH1, then at least one more instruction will be executed before any interrupt is vectored to; this delay guarantees that changes of the interrupt status can be observed by the CPU.

The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at phase 2 of the previous machine cycle. Note that if any interrupt flag is active but its node interrupt request was not responded to for one of the conditions already mentioned, and if the flag is no longer active at a later time when servicing the interrupt node, the corresponding interrupt source will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle interrogates only the pending interrupt requests.

The processor acknowledges an interrupt request by executing a hardware generated LCALL to the corresponding service routine. In some cases, hardware also clears the flag that generated the interrupt, while in other cases, the flag must be cleared by the user's software. The hardware-generated LCALL pushes the contents of the Program Counter (PC) onto the stack (but it does not save the PSW) and reloads the PC with an address that depends on the interrupt node being vectored to, as shown in the [Table 9-1](#).

Program execution returns to the next instruction after calling the interrupt when the RETI instruction is encountered. The RETI instruction informs the processor that the interrupt routine is no longer in progress, then pops the two top bytes from the stack and reloads the PC. Execution of the interrupted program continues from the point where it was stopped. Note that the RETI instruction is important because it informs the processor that the program has left the current interrupt priority level. A simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system on the assumption that an interrupt was still in progress. In this case, no interrupt of the same or lower priority level would be acknowledged.

9.4 Interrupt Response Time

Due to an interrupt event of (the various sources of) an interrupt node, its corresponding request signal will be sampled active at phase 2 in every machine cycle. The value is not polled by the circuitry until the next machine cycle. If the request is active and conditions

Interrupt System

are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two machine cycles. Thus, a minimum of three complete machine cycles will elapse from activation of the interrupt request to the beginning of execution of the first instruction of the service routine as shown in **Figure 9-9**.

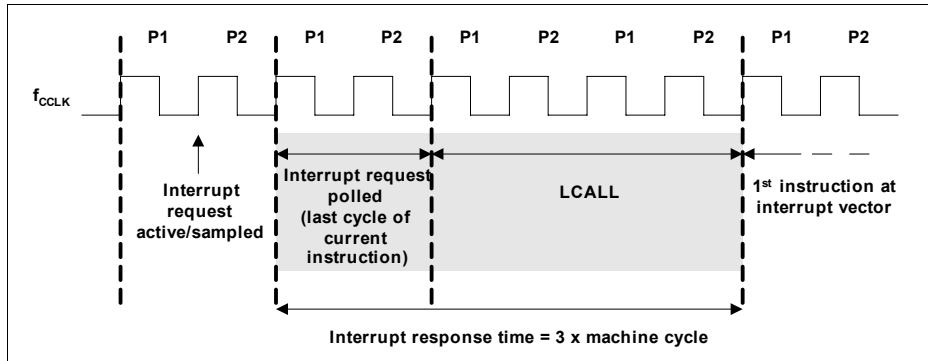


Figure 9-9 Minimum Interrupt Response Time

A longer response time would be obtained if the request is blocked by one of the three previously listed conditions:

1. If an interrupt of equal or higher priority is already in progress, the additional wait time will depend on the nature of the other interrupt's service routine.
2. If the instruction in progress is not in its final cycle, the additional wait time cannot be more than three machine cycles since the longest instructions (MUL and DIV) are only four machine cycles long. See **Figure 9-10**.
3. If the instruction in progress is RETI or a write access to registers IEN0, IEN1 or IP(H), IP1(H), the additional wait time cannot be more than five cycles (a maximum of one more machine cycle to complete the instruction in progress, plus four machine cycles to complete the next instruction, if the instruction is MUL or DIV). See **Figure 9-11**.

Interrupt System

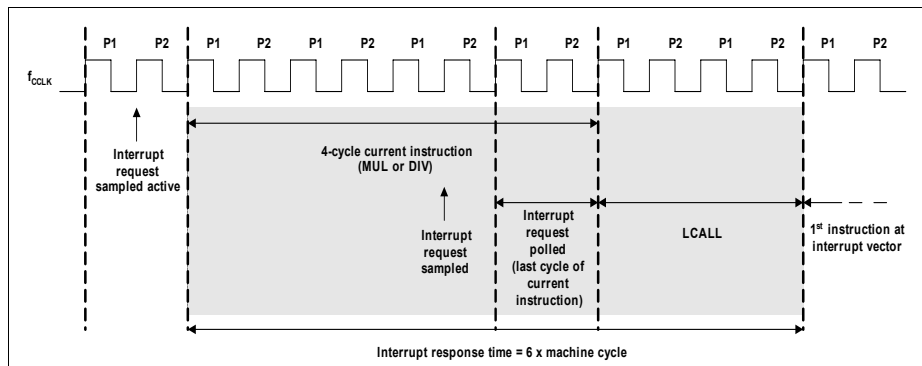


Figure 9-10 Interrupt Response Time for Condition 2

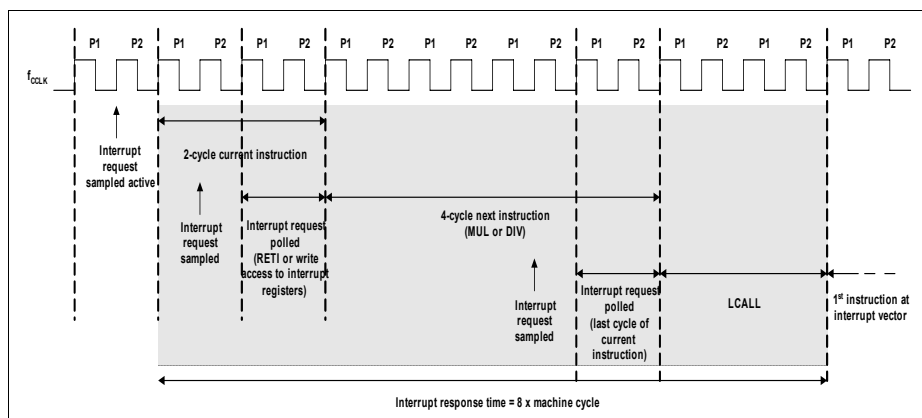


Figure 9-11 Interrupt Response Time for Condition 3

Thus in a single interrupt system, the response time is always more than three machine cycles and less than nine machine cycles (wait states are not considered). When considering wait states, the interrupt response time will be extended depending on the user instructions (also the hardware generated LCALL) being executed during the interrupt response time (shaded region in [Figure 9-10](#) and [Figure 9-11](#)).

9.5 Registers Description

Interrupt Special Function Registers or bits are used for interrupt configuration such as node enable, external interrupt control, interrupt flags and interrupt priority setting.

Table 9-3 lists the SFRs and corresponding address.

Table 9-3 Register Map

Address	Register
RMAP = 0 or 1	
A8 _H	IEN0
E8 _H	IEN1
B8 _H	IP
B9 _H	IPH
F8 _H	IP1
F9 _H	IPH1
88 _H	TCON
98 _H	SCON
RMAP = 0	
EE _H (SCU page 0)	NMICON
EF _H (SCU page 0)	EXICON0
F4 _H (SCU page 0)	EXICON1
F4 _H (SCU page 3)	MODPISEL1
F2 _H (SCU page 0)	IRCON0
F3 _H (SCU page 0)	IRCON1
F5 _H (SCU page 0)	IRCON2
F6 _H (SCU page 0)	IRCON3
F7 _H (SCU page 0)	NMISR

Interrupt System

9.5.1 Interrupt Node Enable Registers

Each interrupt node can be individually enabled or disabled by setting or clearing the corresponding bit in the interrupt enable registers IEN0 and IEN1. Register IEN0 also contains the global interrupt masking bit (EA), which can be cleared to block all pending interrupt requests at once.

The NMI interrupt vector is shared by a number of sources, each of which can be enabled or disabled individually via register NMICON.

After reset, the enable bits in IEN0, IEN1 and NMICON are cleared to 0. This implies that all interrupt nodes are disabled by default.

IEN0

Interrupt Enable Register 0

(A8_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
EA	0	ET2	ES	ET1	EX1	ET0	EX0
rw	r	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
EX0	0	rw	Interrupt Node XINTR0 Enable 0 XINTR0 is disabled 1 XINTR0 is enabled
ET0	1	rw	Interrupt Node XINTR1 Enable 0 XINTR1 is disabled 1 XINTR1 is enabled
EX1	2	rw	Interrupt Node XINTR2 Enable 0 XINTR2 is disabled 1 XINTR2 is enabled
ET1	3	rw	Interrupt Node XINTR3 Enable 0 XINTR3 is disabled 1 XINTR3 is enabled
ES	4	rw	Interrupt Node XINTR4 Enable 0 XINTR4 is disabled 1 XINTR4 is enabled
ET2	5	rw	Interrupt Node XINTR5 Enable 0 XINTR5 is disabled 1 XINTR5 is enabled

Interrupt System

Field	Bits	Type	Description
EA	7	rw	Global Interrupt Mask 0 All pending interrupt requests (except NMI) are blocked from the core. 1 Pending interrupt requests are not blocked from the core.
0	6	r	Reserved Returns 0 if read; should be written with 0.

IEN1

Interrupt Enable Register 1

(E8_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
ECCIP3	ECCIP2	ECCIP1	ECCIP0	EXM	EFTO	ESSC	EADC
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
EADC	0	rw	Interrupt Node XINTR6 Enable 0 XINTR6 is disabled 1 XINTR6 is enabled
ESSC	1	rw	Interrupt Node XINTR7 Enable 0 XINTR7 is disabled 1 XINTR7 is enabled
EFTO	2	rw	Interrupt Node XINTR8 Enable 0 XINTR8 is disabled 1 XINTR8 is enabled
EXM	3	rw	Interrupt Node XINTR9 Enable 0 XINTR9 is disabled 1 XINTR9 is enabled
ECCIP0	4	rw	Interrupt Node XINTR10 Enable 0 XINTR10 is disabled 1 XINTR10 is enabled
ECCIP1	5	rw	Interrupt Node XINTR11 Enable 0 XINTR11 is disabled 1 XINTR11 is enabled

Interrupt System

Field	Bits	Type	Description
ECCIP2	6	rw	Interrupt Node XINTR12 Enable 0 XINTR12 is disabled 1 XINTR12 is enabled
ECCIP3	7	rw	Interrupt Node XINTR13 Enable 0 XINTR13 is disabled 1 XINTR13 is enabled

The bit field PAGE of SCU_PAGE register must be programmed before accessing the NMICON register.

NMICON

NMI Control Register

(EE_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	NMIECC	NMIVDDP	NMIVDDC	NMIOCDS	NMI-FLASH	NMI-OSCCLK	NMIWDT
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NMIWDT	0	rw	Watchdog Timer NMI Enable 0 WDT NMI is disabled. 1 WDT NMI is enabled.
NMIOSCCLK	1	rw	Loss of 48 MHz or 75 KHz Oscillator Clock NMI Enable 0 Loss of 48 MHz or 75 KHz Oscillator Clock NMI is disabled. 1 Loss of 48 MHz or 75 KHz Oscillator Clock NMI is enabled.
NMIFLASH	2	rw	Flash NMI Enable 0 Flash Timer NMI is disabled. 1 Flash Timer NMI is enabled.
NMIOCDS	3	rw	OCDS NMI Enable 0 OCDS NMI is disabled. 1 OCDS NMI is enabled.
NMIVDDC	4	rw	VDDC Prewarning NMI Enable 0 VDDC prewarning NMI is disabled. 1 VDDC prewarning NMI is enabled.

Interrupt System

Field	Bits	Type	Description
NMIVDDP	5	rw	VDDP Prewarning NMI Enable 0 VDDP prewarning NMI is disabled. 1 VDDP prewarning NMI is enabled.
NMIECC	6	rw	ECC Error NMI Enable 0 ECC Error NMI is disabled. 1 ECC Error NMI is enabled.
0	7	r	Reserved Returns 0 if read; should be written with 0.

Note: When the external power supply is 3.3 V, the user must disable NMIVDDP.

9.5.2 External Interrupt Control Registers

The seven external interrupts, EXT_INT[6:0], are driven into the XC82x from the ports. External interrupts can be positive, negative or double edge triggered. Registers EXICON0 and EXICON1 specify the active edge for the external interrupt. Among the external interrupts, external interrupt 0 and external interrupt 1 can be selected to bypass edge detection in the SCU, for direct feed-through to the core. This signal to the core can be further programmed to either low-level or negative transition activated, by the bits IT0 and IT1 in the TCON register. However for edge detection in SCU, TCON.IT0/1 must be set to falling edge triggered. An active edge event detected in SCU will generate internally two CCLK cycle low pulse for detection by core.

If the external interrupt is positive (negative) edge triggered, the external source must hold the request pin low (high) for at least one CCLK cycle, and then hold it high (low) for at least one CCLK cycle to ensure that the transition is recognized. If edge detection is bypassed for external interrupt 0 and external interrupt 1, the external source must hold the request pin "high" or "low" for at least two CCLK cycles.

External interrupts 2 to 6 share their interrupt node with other interrupt sources. Therefore in addition to the corresponding interrupt node enable, each external interrupt 2 to 6 may be disabled individually, and are disabled by default after reset.

The bit field PAGE of SCU_PAGE register must be programmed before accessing EXICON0, EXICON1 and MODPISEL1 registers.

Note: Several external interrupts support alternative input pin. When switching inputs, the active edge/level trigger select and the level on the associated pins should be considered to prevent unintentional interrupt generation.

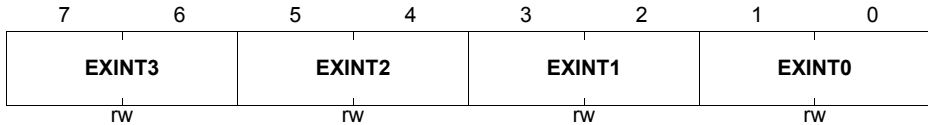
Interrupt System

EXICON0

External Interrupt Control Register 0 (EF_H)

Reset Value: F0_H

RMAP: 0, PAGE: 0



Field	Bits	Type	Description
EXINT0	1:0	rw	External Interrupt 0 Trigger Select 00 Interrupt on falling edge. 01 Interrupt on rising edge. 10 Interrupt on both rising and falling edge. 11 Bypass the edge detection in SCU. The input signal directly feeds to the core.
EXINT1	3:2	rw	External Interrupt 1 Trigger Select 00 Interrupt on falling edge. 01 Interrupt on rising edge. 10 Interrupt on both rising and falling edge. 11 Bypass the edge detection in SCU. The input signal directly feeds to the core.
EXINT2	5:4	rw	External Interrupt 2 Trigger Select 00 Interrupt on falling edge. 01 Interrupt on rising edge. 10 Interrupt on both rising and falling edge. 11 External interrupt 2 is disabled.
EXINT3	7:6	rw	External Interrupt 3 Trigger Select 00 Interrupt on falling edge. 01 Interrupt on rising edge. 10 Interrupt on both rising and falling edge. 11 External interrupt 3 is disabled.

Interrupt System

EXICON1

External Interrupt Control Register 1 (F4_H)

Reset Value: 3F_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	EXINT6	EXINT5	EXINT4				
r	rw	rw	rw				

Field	Bits	Type	Description
EXINT4	1:0	rw	External Interrupt 4 Trigger Select 00 Interrupt on falling edge. 01 Interrupt on rising edge. 10 Interrupt on both rising and falling edge. 11 External interrupt 4 is disabled.
EXINT5	3:2	rw	External Interrupt 5 Trigger Select 00 Interrupt on falling edge. 01 Interrupt on rising edge. 10 Interrupt on both rising and falling edge. 11 External interrupt 5 is disabled.
EXINT6	5:4	rw	External Interrupt 6 Trigger Select 00 Interrupt on falling edge. 01 Interrupt on rising edge. 10 Interrupt on both rising and falling edge. 11 External interrupt 6 is disabled.
0	7:6	r	Reserved Returns 0 if read; should be written with 0.

MODPISEL1

Peripheral Input Select Register 1 (F4_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	EXINT1IS	0	EXINT0IS	0	0	URRIS	
r	rw	r	rw	r	r	rw	

Interrupt System

Field	Bits	Type	Description
EXINT0IS	[4:3]	rw	External Interrupt 0 Input Selection 00 External Interrupt Input EXINT0_0 is selected. 01 External Interrupt Input EXINT0_1 is selected. 10 External Interrupt Input EXINT0_2 is selected. 11 External Interrupt Input EXINT0_3 is selected.
EXINT1IS	6	rw	External Interrupt 1 Input Select 0 External Interrupt Input EXINT1_0 is selected. 1 External Interrupt Input EXINT1_1 is selected.
0	2, 5, 7	r	Reserved Returns 0 if read; should be written with 0.

TCON

Timer and Counter Control/Status Register(88_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
IT0	0	rw	External Interrupt 0 Level/Edge Trigger Control 0 Low level triggered external interrupt 0 is selected 1 Falling edge triggered external interrupt 0 is selected
IT1	2	rw	External Interrupt 1 Level/Edge Trigger Control 0 Low level triggered external interrupt 1 is selected 1 Falling edge triggered external interrupt 1 is selected

9.5.3 Interrupt Flag Registers

The interrupt flags for the different interrupt sources are located in several Special Function Registers. In case of software and hardware access to a flag bit at the same

Interrupt System

time, hardware will have higher priority. The bit field PAGE of SCU_PAGE register must be programmed before accessing IRCONx and NMISR register.

IRCON0

Interrupt Request Register 0
RMAP: 0, PAGE: 0

(F2_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
0	EXINT6	EXINT5	EXINT4	EXINT3	EXINT2	0	
r	rwh	rwh	rwh	rwh	rwh	r	

Field	Bits	Type	Description
EXINTx (x = 2 - 6)	[6:2]	rwh	Interrupt Flag for External Interrupt x This bit is set by hardware and can only be cleared by software. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
0	[1:0], 7	r	Reserved Returns 0 if read; Should be written with 0.

IRCON1

Interrupt Request Register 1
RMAP: 0, PAGE: 0

(F3_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
0			ADCSR1	ADCSR0	RIR	TIR	EIR
r			rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
EIR	0	rwh	Error Interrupt Flag for SSC This bit is set by hardware and can only be cleared by software. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

Interrupt System

Field	Bits	Type	Description
TIR	1	rwh	Transmit Interrupt Flag for SSC This bit is set by hardware and can only be cleared by software. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
RIR	2	rwh	Receive Interrupt Flag for SSC This bit is set by hardware and can only be cleared by software. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
ADCSR0	3	rwh	Interrupt Flag 0 for ADC This bit is set by hardware and can only be cleared by software. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
ADCSR1	4	rwh	Interrupt Flag 1 for ADC This bit is set by hardware and can only be cleared by software. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
0	[7:5]	r	Reserved Returns 0 if read; should be written with 0.

IRCON2

Interrupt Request Register 2

(F5_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
	0		CCU6SR1		0		CCU6SR0
	r		rwh		r		rwh

Field	Bits	Type	Description
CCU6SR0	0	rwh	Interrupt Flag 0 for CCU6 This bit is set by hardware and can only be cleared by software. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

Interrupt System

Field	Bits	Type	Description
CCU6SR1	4	rwh	Interrupt Flag 1 for CCU6 This bit is set by hardware and can only be cleared by software. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
0	[3:1], [7:5]	r	Reserved Returns 0 if read; should be written with 0.

IRCON3

Interrupt Request Register 3

(F6_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
	0		CCU6SR3		0		CCU6SR2
	r		rwh		r		rwh

Field	Bits	Type	Description
CCU6SR3	4	rwh	Interrupt Flag 3 for CCU6 This bit is set by hardware and can only be cleared by software. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
CCU6SR2	0	rwh	Interrupt Flag 2 for CCU6 This bit is set by hardware and can only be cleared by software. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
0	[3:1], [7:5]	r	Reserved Returns 0 if read; should be written with 0.

Interrupt System

TCON

Timer and Counter Control/Status Register(88_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
rwh	rw	rwh	rw	rwh	rw	rwh	rw

Field	Bits	Type	Description
IE0	1	rwh	External Interrupt 0 Flag Set by hardware when external interrupt 0 event is detected. Cleared by hardware when the processor vectors to interrupt routine. Can also be cleared by software.
IE1	3	rwh	External Interrupt 1 Flag Set by hardware when external interrupt 1 event is detected. Cleared by hardware when the processor vectors to interrupt routine. Can also be cleared by software.
TF0	5	rwh	Timer 0 Overflow Flag Set by hardware on Timer/Counter 0 overflow. Cleared by hardware when processor vectors to interrupt routine. Can also be cleared by software.
TF1	7	rwh	Timer 1 Overflow Flag Set by hardware on Timer/Counter 1 overflow. Cleared by hardware when processor vectors to interrupt routine. Can also be cleared by software.

SCON

UART Control/Status Register

(98_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI
rw	rw	rw	rw	rw	rwh	rwh	rwh

Interrupt System

Field	Bits	Type	Description
RI	0	rwh	Serial Interface Receiver Interrupt Flag Set by hardware if a serial data byte has been received. Must be cleared by software.
TI	1	rwh	Serial Interface Transmitter Interrupt Flag Set by hardware at the end of a serial data transmission. Must be cleared by software.

NMISR

NMI Status Register

(F7_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	FNMI ECC	FNMI VDDP	FNMI VDDC	FNMI OCDS	FNMI FLASH	FNMI OSCCLK	FNMI WDT
r	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
FNMIWDT	0	rwh	Watchdog Timer NMI Flag 0 No watchdog NMI has occurred. 1 WDT prewarning has occurred.
FNMIOSCCLK	1	rwh	48 MHz or 75 KHz Oscillator Clock NMI Flag 0 No 48 MHz or 75 KHz Oscillator NMI has occurred. 1 48 MHz or 75 KHz loss of clock has occurred.
FNMIFLASH	2	rwh	Flash Operation Complete NMI Flag 0 No Flash NMI has occurred. 1 Flash operation complete event has occurred.
FNMIOCDS	3	rwh	OCDS NMI Flag 0 No OCDS NMI has occurred. 1 JTAG-receiving or user interrupt request in Monitor Mode has occurred.
FNMI VDDC	4	rwh	VDDC Prewarning NMI Flag 0 No V _{DDC} NMI has occurred. 1 V _{DDC} prewarning (drop below 2.3V) has occurred.

Interrupt System

Field	Bits	Type	Description
FNMIVDDP	5	rwh	VDDP Prewarning NMI Flag 0 No V_{DDP} NMI has occurred. 1 V_{DDP} prewarning (drop below 4.0V for external power supply of 5.0V) has occurred.
FNMIECC	6	rwh	ECC NMI Flag 0 No ECC error has occurred. 1 ECC error has occurred.
0	7	r	Reserved Returns 0 if read; should be written with 0.

Note: NMISR register can only be cleared by software or reset to the default value after the Power-On Reset. Its value is retained on any other resets. This allows the cause of the previous NMI to be checked.

9.5.4 Interrupt Priority Registers

Each interrupt node can be individually programmed to one of the four priority levels available. Two pairs of Interrupt Priority Registers are available to program the priority level of the each interrupt vector. The first pair of Interrupt Priority Registers are SFRs IP and IPH. The second pair of Interrupt Priority Registers are SFRs IP1 and IPH1.

The corresponding bits in each pair of Interrupt Priority Registers select one of the four priority levels as shown in [Table 9-4](#).

Table 9-4 Interrupt Priority Level Selection

IPH.x / IPH1.x	IP.x / IP1.x	Priority Level
0	0	Level 0 (lowest)
0	1	Level 1
1	0	Level 2
1	1	Level 3 (highest)

Note: NMI always has the highest priority (above Level 3); it does not use the priority level selection shown in [Table 9-4](#).

IP

Interrupt Priority Register

(B8_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
0	PT2	PS	PT1	PX1	PT0	PX0	
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PX0	0	rw	Priority Level Low Bit for Interrupt Node XINTR0
PT0	1	rw	Priority Level Low Bit for Interrupt Node XINTR1
PX1	2	rw	Priority Level Low Bit for Interrupt Node XINTR2
PT1	3	rw	Priority Level Low Bit for Interrupt Node XINTR3
PS	4	rw	Priority Level Low Bit for Interrupt Node XINTR4
PT2	5	rw	Priority Level Low Bit for Interrupt Node XINTR5
0	[7:6]	r	Reserved Returns 0 if read; should be written with 0.

Interrupt System

IPH

Interrupt Priority High Register (B9_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
0	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PX0H	0	rw	Priority Level High Bit for Interrupt Node XINTR0
PT0H	1	rw	Priority Level High Bit for Interrupt Node XINTR1
PX1H	2	rw	Priority Level High Bit for Interrupt Node XINTR2
PT1H	3	rw	Priority Level High Bit for Interrupt Node XINTR3
PSH	4	rw	Priority Level High Bit for Interrupt Node XINTR4
PT2H	5	rw	Priority Level High Bit for Interrupt Node XINTR5
0	[7:6]	r	Reserved Returns 0 if read; should be written with 0.

IP1

Interrupt Priority 1 Register (F8_H)

Reset Value: 04_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
PCCIP3	PCCIP2	PCCIP1	PCCIP0	PXM	PFTO	PSSC	PADC
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PADC	0	rw	Priority Level Low Bit for Interrupt Node XINTR6
PSSC	1	rw	Priority Level Low Bit for Interrupt Node XINTR7
PFTO	2	rw	Priority Level Low Bit for Interrupt Node XINTR8
PXM	3	rw	Priority Level Low Bit for Interrupt Node XINTR9
PCCIP0	4	rw	Priority Level Low Bit for Interrupt Node XINTR10
PCCIP1	5	rw	Priority Level Low Bit for Interrupt Node XINTR11

Interrupt System

Field	Bits	Type	Description
PCCIP2	6	rw	Priority Level Low Bit for Interrupt Node XINTR12
PCCIP3	7	rw	Priority Level Low Bit for Interrupt Node XINTR13

IPH1

Interrupt Priority 1 High Register (F9_H)

Reset Value: 04_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
PCCIP3H	PCCIP2H	PCCIP1H	PCCIP0H	PXMH	PFTOH	PSSCH	PADCH
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PADCH	0	rw	Priority Level High Bit for Interrupt Node XINTR6
PSSCH	1	rw	Priority Level High Bit for Interrupt Node XINTR7
PFTOH	2	rw	Priority Level High Bit for Interrupt Node XINTR8
PXMH	3	rw	Priority Level High Bit for Interrupt Node XINTR9
PCCIP0H	4	rw	Priority Level High Bit for Interrupt Node XINTR10
PCCIP1H	5	rw	Priority Level High Bit for Interrupt Node XINTR11
PCCIP2H	6	rw	Priority Level High Bit for Interrupt Node XINTR12
PCCIP3H	7	rw	Priority Level High Bit for Interrupt Node XINTR13

9.6 Interrupt Flag Overview

The interrupt events have interrupt flags that are located in different SFRs. [Table 9-5](#) shows the corresponding SFR to which each interrupt flag belongs. Detailed information on the interrupt flags is provided in the respective peripheral chapters.

Table 9-5 Location of the Interrupt Flags

Interrupt Event	Interrupt Flag	SFR
Timer 0 Overflow	TF0	TCON
Timer 1 Overflow	TF1	TCON

Interrupt System
Table 9-5 Location of the Interrupt Flags (cont'd)

Interrupt Event	Interrupt Flag	SFR
Timer2 Overflow	TF2	T2_T2CON
Timer2 External Event	EXF2	T2_T2CON
UART Receive	RI	SCON
UART Transmit	TI	SCON
LIN End of Synch Byte	EOFSYN	LINST
LIN Synch Byte Error	ERRSYN	LINST
External Interrupt 0	IE0	TCON
External Interrupt 1	IE1	TCON
External Interrupt 2	EXINT2	IRCON0
External Interrupt 3	EXINT3	IRCON0
External Interrupt 4	EXINT4	IRCON0
External Interrupt 5	EXINT5	IRCON0
External Interrupt 6	EXINT6	IRCON0
RTC compare/wakeup interrupt	CFRTC	RTCON
RTC second time interrupt	SFRTC	RTCON
A/D Converter Service Request 0	ADCSR0	IRCON1
A/D Converter Service Request 1	ADCSR1	IRCON1
MDU Result Ready	IRDY	MDUSTAT
MDU Error	IERR	MDUSTAT
SSC Error	EIR	IRCON1
SSC Transmit	TIR	IRCON1
SSC Receive	RIR	IRCON1
IIC Interrupt	IFLG	CNTR
CCU6 Node 0 Interrupt	CCU6SR0 ¹⁾	IRCON2
CCU6 Node 1 Interrupt	CCU6SR1 ¹⁾	IRCON2
CCU6 Node 2 Interrupt	CCU6SR2 ¹⁾	IRCON3
CCU6 Node 3 Interrupt	CCU6SR3 ¹⁾	IRCON3
LEDTSU Time Slice Interrupt	TSF	GLOBCTL1
LEDTSU Time Frame Interrupt	TFF	GLOBCTL1
Watchdog Timer NMI	FNMIWDT	NMISR

Interrupt System
Table 9-5 Location of the Interrupt Flags (cont'd)

Interrupt Event	Interrupt Flag	SFR
48 MHz and 75 KHz Oscillator NMI	FNMIOSCCLK	NMISR
32.768 KHz XTAL Oscillator NMI	FNMIXTALCLK	NMISR
Flash Operation Complete NMI	FNMIFLASH	NMISR
OCDS NMI	FNMIOCDS	NMISR
VDDC Prewarning NMI	FNMIVDDC	NMISR
VDDP Prewarning NMI	FNMIVDDP	NMISR
Flash ECC NMI	FNMIECC	NMISR

1) Each CCU6 interrupt can be assigned to any of the CCU6 interrupt nodes [3:0] via CCU6 registers INPL/INPH.

10 Debug System

This chapter describes the XC800 debug system, in particular focus on the OCDS unit which provides the hardware to support debug functionality.

10.1 Overview

The debug system comprises of the On-Chip Debug Support (OCDS) unit and Debug-Monitor in ROM which provides basic debug functionality for XC800-based systems, controlled directly by an external tool via debug interface pin(s).

Features

The main debug functionality supported are:

- Set breakpoints on instruction address and on address range within the Program Memory
- Set breakpoints on Internal RAM address range
- Support unlimited software breakpoints in Flash/RAM code region
- Process external breaks via debug interface
- Step through the program code
- User handling of OCDS NMI in case of IRAM read/write breakpoint

10.1.1 Components of the Debug System

The components of the debug system are briefly described here.

Debug Interface

The debug interface allows to access the device, such as for data read or write. The debug interface supported:

- Single-pin DAP (SPD)
- 1- or 2-pin UART

The SPD interface refers to the single-pin Device Access Port standardized for the Infineon microcontrollers. It utilizes only one pin DAP1 for serial data in/out.

The UART is a standard interface, however OCDS hardware commands are not supported.

Monitor Program

Part of the Boot ROM is occupied by the OCDS Monitor program (or OCDS firmware). On the command from the debugger, the Monitor program executes the actual instructions for accessing the addressable locations such as memories, sfrs of the device.

On-Chip Debug System Unit (OCDS)

The OCDS hardware is the core of the debug system, controlling the debugging with interfaces to the XC800 CPU.

10.2 Product Specific Information

This section provides product specific information relevant to the OCDS.

10.2.1 Pinning

Figure 10-1 describes the debug pin function in XC82x.

Table 10-1 XC82x Pin Functions and Selection

Pin	Function	Description	Selected By
P0.6	SPD_0	Single-pin debug access port: data	Selectable via BMI value. Refer to Chapter 5 for more details.
P1.0	SPD_1	Single-pin debug access port: data	

10.2.2 Clocking Configuration

The OCDS runs at the CPU operating frequency of either 8 MHz or 24 MHz.

10.2.3 Interrupt Events and Assignment

Instead of entering Monitor Mode on a break event, it can be configured such that OCDS NMI is vectored to. In this configuration, the action on OCDS NMI is fully under user software control. Details of this usage is described in later sections.

Table 10-2 lists the interrupt event sources from the OCDS, and the corresponding event interrupt enable bit and flag bit.

Table 10-2 OCDS Interrupt Events

Event	Event Interrupt Enable Bit	Event Flag Bit
IRAM read event	MMICR.NMIRRE	MMICR.FNMIRR
IRAM write event	MMICR.NMIRWE	MMICR.FNMIRW

Table 10-3 shows the interrupt node assignment for each OCDS interrupt source.

Table 10-3 OCDS Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
IRAM read event	NMICON.NMIOCDS	NMISR.FNMIOCDS	73 _H
IRAM write event			

10.2.4 Debug Suspend Control

It is enabled for selected modules to suspend operation when Monitor Mode becomes active. The module functions that can be enabled for debug suspend are:

- Watchdog timer
- Timer 12 of CCU6
- Timer 13 of CCU6
- Timer 2
- RTC timer
- LEDTSCU counters

By debug suspend, these module functions are frozen during the duration of the device in Monitor Mode, e.g. timers stop counting. Register MODSUSP is used to control the debug suspend function. The bit field of SCU_PAGE register must be programmed before accessing the MODSUSP register.

MODSUSP

Module Suspend Control Register (F6_H)

Reset Value: 01_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	LTSSUSP	RTCSUSP	T2SUSP	T13SUSP	T12SUSP	WDTSUSP	
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
WDTSUSP	0	rw	SCU Watchdog Timer Debug Suspend Bit 0 WDT will not be suspended. 1 WDT will be suspended.

Debug System

Field	Bits	Type	Description
T12SUSP	1	rw	Timer 12 Debug Suspend Bit 0 Timer 12 in Capture/Compare Unit will not be suspended. 1 Timer 12 in Capture/Compare Unit will be suspended. In addition, the T12 PWM outputs are set to the inactive level and capture inputs are disabled when suspended.
T13SUSP	2	rw	Timer 13 Debug Suspend Bit 0 Timer 13 in Capture/Compare Unit will not be suspended. 1 Timer 13 in Capture/Compare Unit will be suspended. In addition, the T13 PWM outputs are set to the inactive level and capture inputs are disabled when suspended.
T2SUSP	3	rw	Timer2 Debug Suspend Bit 0 Timer2 will not be suspended. 1 Timer2 will be suspended.
RTCSUSP	4	rw	Real-time Clock Debug Suspend Bit 0 Real-time clock will not be suspended. 1 Real-time clock will be suspended.
LTSSUSP	5	rw	LED and Touch-sense Counters Debug Suspend Bit 0 LED and Touch-sense Counters will not be suspended. 1 LED and Touch-sense Counters will be suspended.
0	[7:6]	r	Reserved Returns 0 if read; should be written with 0.

10.2.5 JTAG ID

The JTAG ID for the XC82x devices is given in [Table 10-4](#).

Table 10-4 JTAG ID Number

Device Type	Device Name	JTAG ID
Flash	XC82x	101BC083 _H

10.3 Functional Overview of the Debug System

XC800 debug-operation is based on close interaction between the OCDS-hardware and the Monitor program in ROM.

These operations are elaborated in the following.

10.3.1 Recognizing Debug-events

The OCDS hardware takes care to startup properly the Monitor, in case some **Debug-event** happens.

The debug events (breakpoints) supported are described in following.

Hardware Breakpoints

Up to 4 hardware breakpoints can be set, causing a break in case of

- code is fetched from a defined Program Memory (IP) address - the break is set just at one instruction from the user software
- code is fetched from a defined Program Memory (IP) address-range - the break is set over a number of consecutive instructions from the user software
- Read or Write (selectable) is performed at a defined Internal RAM address - the break is caused by moving data from/to a memory location

For setting of breakpoint on instruction address,HWBPx defines the 16-bit address. For setting of breakpoint on IRAM address, HWBP2/3L and HWBP2/3H define the 8-bit IRAM address range.

The configurations supported are:

- Breakpoint 0
- Breakpoint 1
 - Two equal breakpoints on
Instruction Address = HWBP0 and
Instruction Address = HWBP1 or
 - One range breakpoint on
HWBP0 <= Instruction Address <= HWBP1
- Breakpoint 2
 - One equal breakpoint on Instruction Address = HWBP2, or
 - One range breakpoint on HWBP2L <= IRAM Read Address <= HWBP2H
- Breakpoint 3
 - One equal breakpoint on Instruction Address = HWBP3, or
 - One range breakpoint on HWBP3L <= IRAM Write Address <= HWBP3H

Setting both values for a range breakpoint to the same address leads to generation of an equal breakpoint.

Software Breakpoints

These are implemented by storing into the code XC800-specific TRAP instruction (not 8051-standard), while at the same time TRAP_EN bit within the Extended Operation (EO) register is also set to 1.

Upon fetching a TRAP instruction, a breakpoint is generated and the relevant Break Action is taken.

The software breakpoints are in fact similar in behavior to the equal breakpoints on Instruction address, except that they are raised by a program code instead of specialized (compare) logic.

An unlimited number of software breakpoints can be set by replacing the original instruction codes into the user program. Of course this is possible only at addresses, where RAM/Flash is implemented.

External Breaks

External breaks could be:

- Requested via the debug interface by OCDS command: An external Debugger can activate the Monitor and interrupt a user program, running on the target XC800.

10.3.2 Activating the Monitor Program

Once a Debug-event is identified and the OCDS-registers are properly programmed, the Monitor program (in ROM) is started as a primary reaction.

The OCDS hardware ensures that the Monitor is always safely started, and fully independent of the current system status at the moment when the debug action is taken. Also, interrupt requests optionally raised during Monitor-entry will not disturb the Monitor firmware functioning.

Once started, the Monitor runs with its own stack- and data- memory, which guarantees that all of the core and memory resources will be found untouched when returning control back to the user program. Therefore the OCDS-debugging is fully non-destructive.

The functions of the Monitor include:

- Communication with an external Debugger via the debug interface
- Read/write access to arbitrary memory locations and Special Function Registers (SFRs), including the Instruction Pointer and password-protected bits
- Configuring OCDS and setting/removing breakpoints
- Executing a single instruction (step-mode)

Note: Details of the Monitor program functionality is not covered in this documentation.

10.3.3 Debug Suspend Control

Next to the basic debug functionality - setting breakpoints and halting the execution of user software - OCDS supports also for module suspend during debugging.

As long as the device is in monitor mode (i.e. while the user software is not running but in break) and if debug suspend functionality is generally enabled by on-chip software, modules or functions can be suspended in this duration.

This feature could be quite useful, especially regarding the Watchdog Timer: it prevents unintentional WDT-resets during the debug session. Usually debug suspend is provided for other timer-modules. Stopping counters provide for a complete "freeze" of the device-status during a break.

Generally, the debug suspend control bits (global enable in OCDS and individual selections in SCU) are disabled after reset, i.e. by default no module will be suspended upon a break. Normally for debugging the device will be started in OCDS mode and then the monitor will be invoked before to start any user code. The debugger should provide for user configuration of suspend-controls.

10.3.4 Running the Monitor

The Monitor runs with its own stack- and data- work space, therefore the complete status of the core is saved and later - restored before returning to the user code.

Once the Monitor is running, it can access for read and write all of the system resources, and data can be communicated with the Debugger.

10.3.5 Returning to the User Program

When a return to User-Program is requested, the Monitor sets core-registers including the Instruction Pointer (IP) as required. This can be either the same status as at the Breakpoint, or including some changes done during the Break by a host debugger. Then a jump to the location pointed by IP is performed and the target system returns to User Mode.

10.3.6 Single Step Execution

For this functionality OCDS operates at first as when Returning to the User Program, but also a dedicated flag is set into a control-register (MMCR.MSTEP). This way, just one user-instruction is executed before re-entering the Monitor again.

10.4 Breakpoint Generation Module

Breakpoint generation is only possible in Monitor Mode.

Attention: *To be able to debug via Monitor Mode, the application software should not change the TRAP_EN bit within Extended Operation (EO) register.*

Breakpoints can be classified into three types:

- **Break Before Make**
The break happens just before the break instruction (i.e. the instruction causing the break) is executed. Therefore, the break instruction itself will be the next instruction from the user program flow but executed only after the relevant debug action has been taken.
- **Break After Make**
The break happens immediately after the instruction causing it has been executed. Therefore, the break instruction itself has already been executed when the relevant debug action is taken.
- **Break Now**
The events of this type are asynchronous to the code execution inside the XC82x and there is no "instruction causing the debug event" in this case. The debug action is performed by OCDS "as soon as possible" once the debug event is raised.

10.4.1 Generating Hardware Breakpoints

This block is built around a set of comparators, dedicated to recognize two types of hardware breakpoints:

10.4.1.1 Breakpoints on Instruction Address

These IP breakpoints are generated, only when the BP-address is matched for the first byte of an instruction - i.e. just for the opcode. In case of 2- and 3- byte instructions, the break will not be generated at all for addresses of the second and third instruction-bytes.

The IP breakpoints are of Break Before Make type, therefore the instruction at the breakpoint is executed only after the proper debug action is taken.

Address Comparators, Control bits and Flags

Below the comparators used for IP Breakpoints are described:

- **CMP0**
A 16-bit comparator between the two **HWBP0** registers (A-side) and the 16-bit Program Memory Address Bus (B-side).
Two output signals are generated:
 - A=B - (break on IP= **HWBP0**)
 - A<B - see below.

- **CMP1**

A 16-bit comparator between the two **HWBP1** registers (A-side) and the 16-bit Program Memory Address Bus (B-side).

Two output signals are generated:

- $A=B$ - (break on $IP=HWBP1$)
- $A \geq B$ when activated together with $A \leq B$ from **CMP0** - (break on **HWBP0** $= IP \leq HWBP1$)

- **CMPIP2**

A 16-bit comparator between the two **HWBP2** registers (A-side) and the 16-bit Program Memory Address Bus (B-side).

One output signal is generated:

- $A=B$ - (break on $IP=HWBP2$)

- **CMPIP3**

A 16-bit comparator between the two **HWBP3** registers (A-side) and the 16-bit Program Memory Address Bus (B-side).

One output signal is generated:

- $A=B$ - (break on $IP=HWBP3$)

10.4.1.2 Breakpoints on Internal RAM Address

These Internal RAM (IRAM) Breakpoints are recognized primary by observing the Internal Data Memory Address Buses, respectively for Breakpoints on Read and Write addresses respectively.

The IRAM breakpoints are of Break After Make type, therefore the proper debug action is taken immediately after the operation to the breakpoint address is performed.

The OCDS supports only range breakpoints on IRAM address.

The OCDS differentiates between a breakpoint on read and a breakpoint on write operation to the IRAM.

The exact processing is different in case of Read and Write Breakpoints.

Address Comparators, Control bits and Flags

Below the comparators used for Memory Access Breakpoints are described:

- **CMPRL2, CMPRH2**

Two 8-bit comparators:

- **CMPRL2** - between **HWBP2L** register (A-side) and the 8-bit IRAM Source Address Bus (B-side), generating $A \leq B$;
- **CMPRH2** - between **HWBP2H** register (A-side) and the 8-bit IRAM Source Address Bus (B-side), generating $A \geq B$.

Then:

break on **HWBP2L** $= IPRAM\ Read\ Address \leq HWBP2H$.

- **CMPWL3, CMPWH3**

Two 8-bit comparators:

- CMPWL3 - between HWBP3L register (A-side) and the 8-bit IRAM Destination Address Bus (B-side), generating $A < B$;
- CMPWH3 - between HWBP3H register (A-side) and the 8-bit IRAM Destination Address Bus (B-side), generating $A \geq B$. Then:
break on **HWBP3L** = < IRAM Write Address <= **HWBP3H**.

10.4.1.3 Tracing changes in Break Address

The mechanism to generate Hardware Breakpoints assures, that "immediately" upon a break-condition match, the XC800 core is forced to Debug Mode. The Program Counter freezes then at a value, related to the address of instruction that would have been fetched, had the core not been in Debug Mode.

When entered, the Monitor will determine the source of the Break and relevant information is to be forwarded to the Debugger, concerning the Break Event source (IRAM read/write)

10.4.2 Processing External Breaks

Possible external break-requests are generated and processed as follows:

1. a request comes from one of the possible source(s)
 - a) Debugger-request via SPD - a dedicated command has been sent, with **MMODE_r=1**, **RRF_r=0** and data-Byte=A5_H, while at the same time:
MMCR2.MMODE=0 and **MMCR2.ALTDI=0**
2. the core ends executing current instruction,
the core enters Debug Mode

Note: External Breaks are not possible inside a NMI-servicing routine.

For more information refer to [Section 10.7](#).

10.4.3 Processing Software Breakpoints

Software breakpoints are of Break Before Make type, therefore the instruction at the breakpoint is executed only after the proper debug action is taken.

Upon fetching a TRAP instruction, Software Breakpoint is effected. Due to this, it goes to Debug Mode.

Attention: The user software must not clear the TRAP_EN bit in EO register.

The OCDS reacts in two ways:

- a Break event is recognized for Entering Monitor Mode
- the Software Break is ignored
This happens when the above condition is not satisfied, i.e. basically when the Software Breakpoints are disabled.

Upon matching a not enabled Software Breakpoint, the user software execution is suppressed for 4 clock cycles after the TRAP instruction.

10.5 Reactions on Breakpoints without Monitor Entry

In case a Break-event happens while Monitor Mode is disabled
, the Monitor program is not started (MM_no_entry) but another action is triggered by OCDS system.

10.5.1 Triggering a NMI request

Instead of starting the Monitor program, OCDS NMI is requested, refer [Section 10.6.1](#).
The flags FNMIRR/FNMIRW here set can be cleared only per software - this must be done by the user NMI-servicing routine.

10.6 NMI Request and Control by OCDS

The OCDS module provides hardware allowing to generate a NMI request itself as well as to control the general NMI-processing within the XC800 system.

10.6.1 NMI request from OCDS

The OCDS generates one interrupt request at the output **ocds_nmi_o**. This signal goes to the Interrupt-Management Module into XC800 SCU (System Control Unit), where it sets a flag within NMI Status Register NMISR. In case OCDS is an enabled source in NMI Control Register NMICON, a non-maskable interrupt request will be activated to the core.

The NMI-request can be activated by OCDS upon memory-access performed to defined address areas (separately configurable for read/write operations) in XC800 Internal RAM. This is an advanced and fully controllable by application code feature, allowing to utilize parts of OCDS not for debugging but to support a general, user-accessible functionality.

The basics for implementation of this feature are:

- the Breakpoint Generation module is able to recognize read/write accesses to IRAM address-ranges (refer to [Section 10.4.1.2](#))
- upon a breakpoint match but while MMCR2.MBCON=1, the Monitor firmware is not started and therefore in fact no debug-session takes place.

To configure, trigger and handle an OCDS-NMI request upon IRAM access (refer also to [Section 10.5.1](#)):

1. at all is possible only as long as MMCR2.MBCON=1
2. user software must execute the following sequence:
 - a) set the NMI-enable bit(s) NMIRRE/NMIRWE in MMICR, respectively for request upon IRAM read/write
 - b) write the low address(es) of the observed area into HWBP2L/HBWP3L, respectively for NMI-request upon IRAM read/write
 - c) write the high address(es) of the observed area into HWBP2H/HBWP3H, respectively for NMI-request upon IRAM read/write
3. after the settings according to point 2) are done and as long as MBCON=1, a NMI-request will be triggered by the OCDS upon an access to the respectively configured (for read/write) IRAM area(s).
4. in case a NMI-request has been raised by the OCDS, respective flag **FNMIIR/FNMIRW** is set in MMICR - refer to the register-description in [Section 10.8.1](#) and functional definition in [Section 10.5.1](#).
If the request is sent to the core - in case enabled by NMICON.NMIOCD=1 (in SCU) - then a **NMI-servicing routine** will be invoked by a jump to its **standard location in XC800 Program Memory**, and this routine:

Debug System

- a) can identify OCDS as NMI-requesting source and the type of IRAM-access causing the request by evaluating NMISR.FNMIOCDs (in SCU) and MMICR.FNMIRR/FNMIRW (in OCDS)
 - b) must clear FNMIRR/FNMIRW flags before its end, to allow a proper reaction on next NMIs.
5. any attempt to configure/activate a Breakpoint for debugging by writing into MMBPCR will reset the internal nmi_r*e/nmi_r*e_l flags and therefore - disable the generation of NMI-request upon IRAM access.

Attention: all the software-handling of OCDS NMI (servicing routine) is to be supported by user code!

10.6.2 General NMI control by OCDS

The controls are aimed to support a correct NMI-related behavior:

- all the pending NMI requests raised before Monitor Mode entry and as long as the system is in Monitor Mode will be not affected, i.e. these requests are safely kept and sent to the core when the system returns to User Mode

10.7 NMI-mode priority over Debug-mode

While the core is in NMI-mode (after an NMI-request has been accepted and before the RETI instruction is executed, i.e. the time during a NMI-servicing routine), certain debug functions are blocked/restricted:

1. No external break (refer to [Section 10.4.2](#)) is possible while the core is servicing a NMI.
External break requested inside a NMI-servicing routine will be taken only after RETI is executed.
2. A breakpoint into NMI-servicing routine is taken, but single stepping is not possible afterwards.
If a step is requested, the servicing routine will run continuously and monitor mode will be invoked again only after a RETI is executed.

Hardware breakpoints and software breakpoints proceed as normal while CPU is in NMI-mode.

10.8 Registers Description

The OCDS Special Function Registers are accessed from the mapped SFR area. [Table 10-5](#) lists the (direct addressable) SFRs and corresponding address.

Table 10-5 Register Map (Direct Addressable)

Address	Register
F4 _H	MMICR
F6 _H	HWBPSR
F7 _H	HWBPDR
EC _H	MMWR2

Additionally, there are Hardware Breakpoint Registers, which are indirect accessible via **HWBPSR** (register select) and **HWBPDR** (data) - refer to [Table 10-6](#) and [Table 10-7](#) (for more information look at [Section 10.8](#)).

Table 10-6 Hardware Breakpoint Registers (Indirect Accessed)

Register	Description
HWBP0L	Hardware Breakpoint 0 Low Register
HWBP0H	Hardware Breakpoint 0 High Register
HWBP1L	Hardware Breakpoint 1 Low Register
HWBP1H	Hardware Breakpoint 1 High Register
HWBP2L	Hardware Breakpoint 2 Low Register
HWBP2H	Hardware Breakpoint 2 High Register
HWBP3L	Hardware Breakpoint 3 Low Register
HWBP3H	Hardware Breakpoint 3 High Register

Table 10-7 HWBPSR [3:0]: Selecting Hardware Breakpoint Registers

BPSEL	Register Selected	BPSEL	Register Selected
0xxx	reserved - no selection		
1000	HWBP0L	1001	HWBP0H
1010	HWBP1L	1011	HWBP1H
1100	HWBP2L	1101	HWBP2H
1110	HWBP3L	1111	HWBP3H

10.8.1 Control and Status Registers

MMICR

Monitor Mode Interrupt Control Register (F4_H)

Reset value: 00_H

RMAP: 1, PAGE: X

7	6	5	4	3	2	1	0
				FNMIW	FNMIW	NMIWE	NMIWE
				rwh	rwh	rw	rw

Field	Bits	Type	Description
NMIWE	0	rw	NMI upon IRAM read (refer to Section 10.6.1): 0 Disabled 1 Enabled
NMIWE	1	rw	NMI upon IRAM write (refer to Section 10.6.1): 0 Disabled 1 Enabled
FNMIW	2	rwh	IRAM Read NMI Flag
FNMIW	3	rwh	IRAM Write NMI Flag

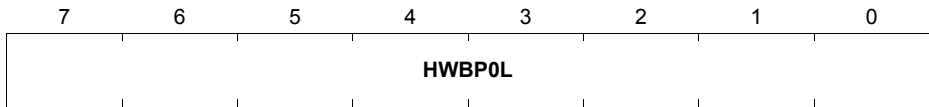
FNMIW / FNMIW bits:

-set by hardware in case of NMI request from OCDS activated upon read/write access to the Internal RAM (refer to [Section 10.5.1](#) and [Section 10.6.1](#))

-the software can only reset these bits

HWBP0L

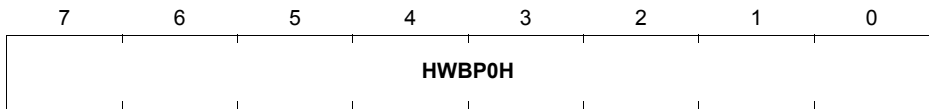
Hardware Breakpoint 0 Low Register written via HWBPDR **Reset value: 00_H**



Field	Bits	Type	Description
HWBP0L	[7:0]		The Low Byte from Compare Address HWBP0

HWBP0H

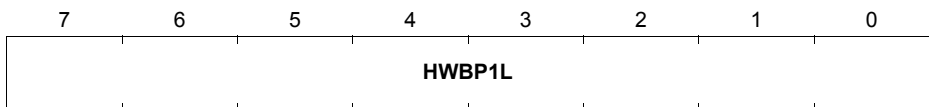
Hardware Breakpoint 0 High Register written via HWBPDR **Reset value: 00_H**



Field	Bits	Type	Description
HWBP0H	[7:0]		The High Byte from Compare Address HWBP0

HWBP1L

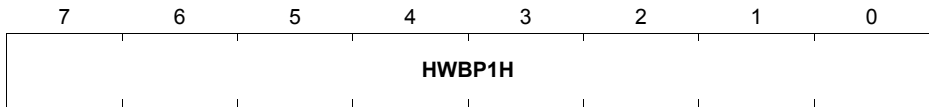
Hardware Breakpoint 1 Low Register written via HWBPDR **Reset value: 00_H**



Field	Bits	Type	Description
HWBP1L	[7:0]		The Low Byte from Compare Address HWBP1

HWBP1H

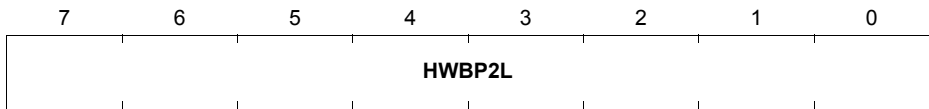
Hardware Breakpoint 1 High Register written via HWBPDR **Reset value: 00_H**



Field	Bits	Type	Description
HWBP1H	[7:0]		The High Byte from Compare Address HWBP1

HWBP2L

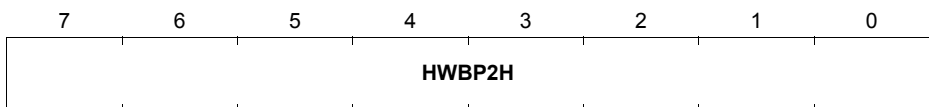
Hardware Breakpoint 2 Low Register written via HWBPDR **Reset value: 00_H**



Field	Bits	Type	Description
HWBP2L	[7:0]		The Low Byte from Compare Address HWBP2

HWBP2H

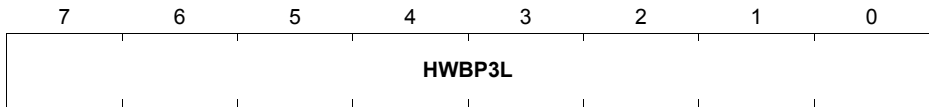
Hardware Breakpoint 2 High Register written via HWBPDR **Reset value: 00_H**



Field	Bits	Type	Description
HWBP2H	[7:0]		The High Byte from Compare Address HWBP2

HWBP3L

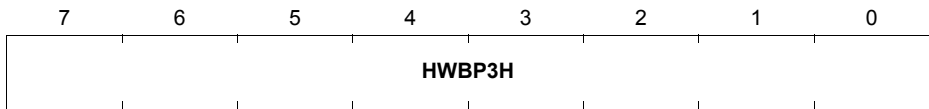
Hardware Breakpoint 3 Low Register written via HWBPDR **Reset value: 00_H**



Field	Bits	Type	Description
HWBP3L	[7:0]		The Low Byte from Compare Address HWBP3

HWBP3H

Hardware Breakpoint 3 High Register written via HWBPDR **Reset value: 00_H**



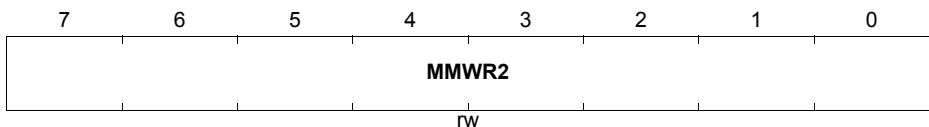
Field	Bits	Type	Description
HWBP3H	[7:0]		The High Byte from Compare Address HWBP3

10.8.3 Monitor Work Register

MMWR2 can be used for general purposes when no debug-session is possible: if the device is not started in OCDS mode and no external device is connected.

MMWR2

Monitor Work Register 1 (**EC_H**) **Reset value: 00_H**
RMAP: 1, PAGE: X



Field	Bits	Type	Description
MMWR2	[7:0]	rw	Work location 2 for the Monitor Program

11 Parallel Ports

XC822 has 13 port pins and XC824 has 17 port pins. Both of them are organized into three parallel ports, Port 0 (P0) to Port 2 (P2). Each pin has a pair of internal pull-up and pull-down devices that can be individually enabled or disabled. Ports P0 and P1 are bidirectional and can be used as general purpose input/output (GPIO) or to perform alternate input/output functions for the on-chip peripherals. When configured as an output, the open drain mode can be selected. Port P2 is an input-only port, providing general purpose input functions, alternate input functions for the on-chip peripherals, and also analog inputs for the Analog-to-Digital Converter (ADC).

Standard Bidirectional Port Features:

- Configurable pin direction
- Configurable pull-up/pull-down devices
- Configurable open drain mode
- Transfer data through digital inputs and outputs (general purpose I/O)
- Alternate input/output for on-chip peripherals
- Input and output drivers are disabled during power save mode

Input Port Features:

- Configurable input driver
- Configurable pull-up/pull-down devices
- Receive data through digital input (general purpose input)
- Alternate input for on-chip peripherals
- Analog input for ADC module
- Input driver is disabled during power save mode

11.1 General Port Operation Description

Figure 11-1 shows the block diagram of an XC82x bidirectional port pin. Each port pin is equipped with a number of control and data bits, thus enabling very flexible usage of the pin. By defining the contents of the control register, each individual pin can be configured as an input or an output. The user can also configure each pin as an open drain pin with or without internal pull-up/pull-down device.

The input and output drivers of the standard bidirectional port are always enabled during normal operation except in power-down mode. However, port pin(s) required as a wake-up source during power-down mode will not be disabled.

In output mode, the output driver drives the value supplied through the multiplexer to the port pin. Each pin in output mode can be switched to open drain mode or normal mode (push-pull mode) via the register Px_OD (x = 0, 1).

The output multiplexer in front of the output driver enables the port output function to be used for different purposes. If the pin is used for general purpose output, the multiplexer

Parallel Ports

is switched by software to the data register Px_DATAOUT. Software can set or clear the bit in Px_DATAOUT and therefore directly influence the state of the port pin. If an on-chip peripheral uses the pin for output signals, alternate output lines (AltDataOut) can be switched via the multiplexer to the output driver circuitry. Selection of the alternate function is defined in registers P0_ALTSEL0, P0_ALTSEL1, P0_ALTSEL2, P1_ALTSEL0 and P1_ALTSEL1.

To configure the pin to input mode (default after reset), the output driver must be switched to high-impedance using the following steps:

- Enable the the open drain function via register Px_OD
- Set output driver to high-impedance by writting '1' to the port pin via register Px_DATAOUT
- Select normal GPIO as alternate output function via register Px_ALTSELx

Setting to normal GPIO alternate function is not necessary if the alternate output line is ensured to held at high during input mode.

The actual voltage level present at the port pin is translated into a logic 0 or 1 via a Schmitt-Trigger device and can be read via the register Px_DATAIN.

Each pin can also be programmed to activate an internal weak pull-up or pull-down device. Register Px_PUDSEL selects whether a pull-up or the pull-down device is activated while register Px_PUDEN enables or disables the pull device.

A following code shows the configuration of all Port 0 pins to input mode.

```
ANL   SYSCON0, #0xFE
MOV   PORT_PAGE, #0x03
MOV   P0_OD, #0xFF           ;Enable open drain
MOV   PORT_PAGE, #0x00
MOV   P0_DATAOUT, #0xFF      ;Set P0_DATAOUT (PORT_PAGE = 0x00)
MOV   PORT_PAGE, #0x02
MOV   P0_ALTSEL0, #0x00      ;Select Alternate function as GPIO
MOV   P0_ALTSEL1, #0x00      ;Px_ALTSELx in PORT_PAGE = 0x02
MOV   P0_ALTSEL2, #0x00
```

Parallel Ports

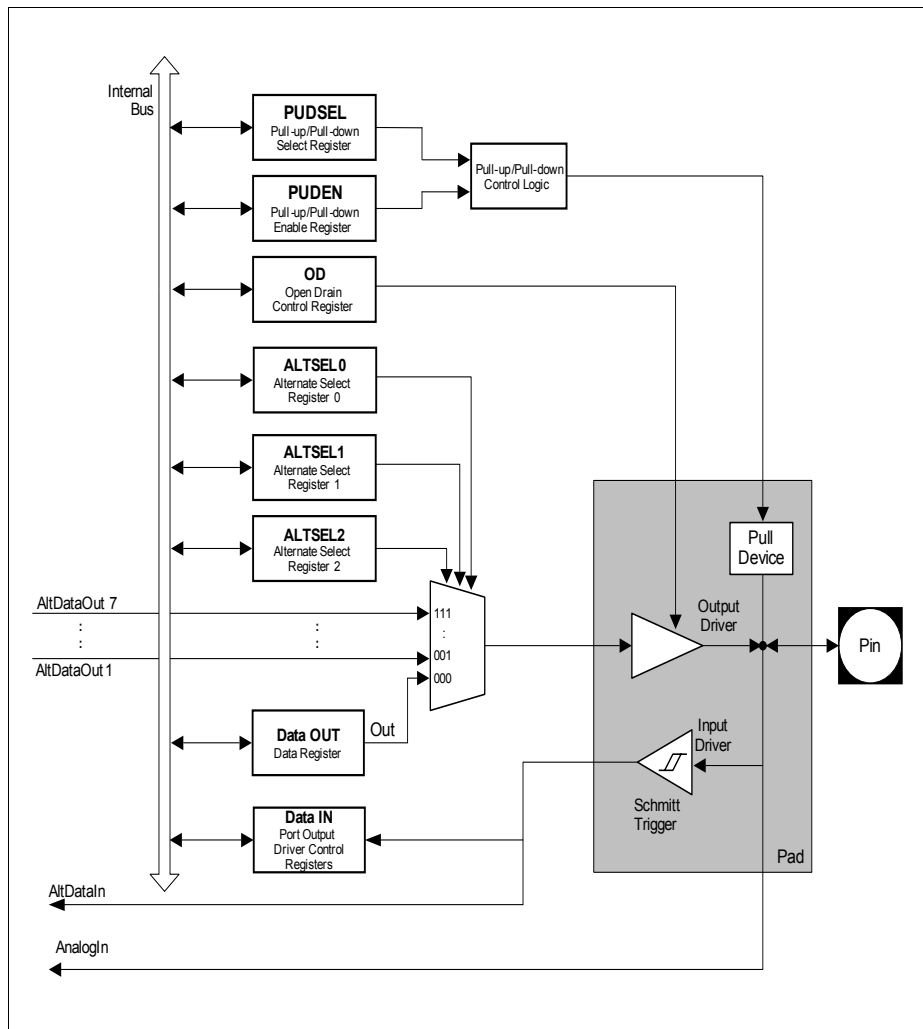


Figure 11-1 General Structure of Standard Bidirectional Port

Parallel Ports

Figure 11-2 shows the structure of an input-only port pin. Each P2 pin can only function in input mode. Register P2_EN is provided to enable or disable the input driver. When the input driver is enabled, the actual voltage level present at the port pin is translated into a logic 0 or 1 via a Schmitt-Trigger device and can be read via the register P2_DATAIN. Each pin can also be programmed to activate an internal weak pull-up or pull-down device. Register P2_PUDSEL selects whether a pull-up or the pull-down device is activated while register P2_PUDEN enables or disables the pull device. The analog input (AnalogIn) bypasses the digital circuitry and Schmitt-Trigger device for direct feed-through to the ADC input channel.

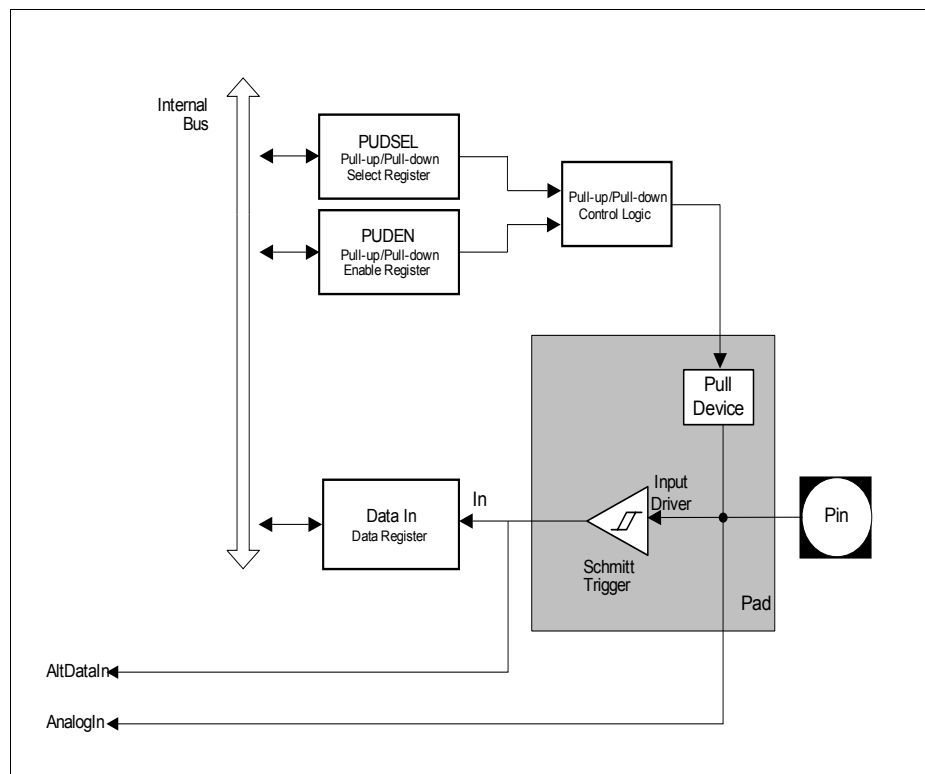


Figure 11-2 General Structure of Input Port

11.1.1 General Register Description

This section describes the SFR registers available on the XC82x module.

11.1.1.1 Register Map

The Port SFRs are located in the standard memory area (RMAP = 0) and are organized into 4 pages. The PORT_PAGE register is located at address 8E_H. It contains the page value and page control information.

The addresses of the Port SFRs are listed in [Table 11-1](#).

Table 11-1 SFR Address List for Pages 0-3

Address	Page 0	Page 1	Page 2	Page 3
80 _H	P0_DATAOUT	P0_PUDSEL	P0_ALTSEL0	P0_OD
86 _H	P0_DATAIN	P0_PUDEN	P0_ALTSEL1	
85 _H			P0_ALTSEL2	
90 _H	P1_DATAOUT	P1_PUDSEL	P1_ALTSEL0	P1_OD
91 _H	P1_DATAIN	P1_PUDEN	P1_ALTSEL1	
92 _H				
93 _H		P2_PUDSEL		
94 _H	P2_DATAIN	P2_PUDEN		P2_EN

PORT_PAGE

Page Register for PORT
RMAP: 0, PAGE: X

(8E_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
OP		STNR		0	PAGE		
w		w		r	rwh		

Field	Bits	Type	Description
PAGE	[2:0]	rwh	Page Bits When written, the value indicates the new page address. When read, the value indicates the currently active page = addr [y:x+1]

Parallel Ports

Field	Bits	Type	Description
STNR	[5:4]	w	Storage Number This number indicates which storage bit field is the target of the operation defined by bit OP. If OP = 10 _B , the contents of PAGE are saved in STx before being overwritten with the new value. If OP = 11 _B , the contents of PAGE are overwritten by the contents of STx. The value written to the bit positions of PAGE is ignored. 00 ST0 is selected. 01 ST1 is selected. 10 ST2 is selected. 11 ST3 is selected.
OP	[7:6]	w	Operation 0X Manual page mode. The value of STNR is ignored and PAGE is directly written. 10 New page programming with automatic page saving. The value written to the bit positions of PAGE is stored. In parallel, the former contents of PAGE are saved in the storage bit field STx indicated by STNR. 11 Automatic restore page action. The value written to the bit positions PAGE is ignored and instead, PAGE is overwritten by the contents of the storage bit field STx indicated by STNR.
0	3	r	Reserved Returns 0 if read; should be written with 0.

11.1.1.2 Register Overview

The individual control and data bits of each parallel port are implemented in a number of 8-bit registers. Bits with the same meaning and function are assembled together in the same register. The registers configure and use the port as general purpose I/O or alternate function input/output.

For port P1 and P2, not all the registers in [Table 11-2](#) are implemented. The availability and definition of registers specific to each port is defined in [Section 11.2](#) to [Section 11.4](#). This section provides only an overview of the different port registers.

Table 11-2 Port Registers

Register Short Name	Register Long Name	Description
Px_DATAOUT	Port x Data Out Register	Page 11-7
Px_DATAIN	Port x Data In Register	Page 11-8
Px_OD	Port x Open Drain Control Register	Page 11-8
Px_PUDSEL	Port x Pull-Up/Pull-Down Select Register	Page 11-9
Px_PUDEN	Port x Pull-Up/Pull-Down Enable Register	Page 11-9
Px_ALTSEL0	Port x Alternate Select Register 0	Page 11-10
Px_ALTSEL1	Port x Alternate Select Register 1	Page 11-10
Px_ALTSEL2	Port x Alternate Select Register 2	Page 11-10
Px_EN	Port x Input Control Register	Page 11-10

Note: Not all the registers are implemented for each port.

Port Data Out Register

If a port pin is used as general purpose output, output data is written into register Px_DATAOUT of port x.

A read operation of Px_DATAOUT returns the register value and not the state of the Px_DATAOUT pins.

Px_DATAOUT

Port x Data Out Register

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 7)	n	rw	Port x Pin n Data Value 0 Port x pin n data value = 0 1 Port x pin n data value = 1 (default)

Parallel Ports
Port Data In Register

If a port pin is used as general purpose input, the value at a port pin can be read through the register Px_DATAIN. The data register Px_DATAIN always contains a latched value of the assigned port pin, even if the port pin is assigned as output.

Px_DATAIN
Port x Data In Register

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
Pn (n = 0 - 7)	n	rh	Port x Pin n Data Value 0 Port x pin n data value = 0 1 Port x pin n data value = 1

Open Drain Control Register

Each pin in output mode can be switched to open drain mode. If driven with 1, no driver will be activated and the pin output state depends on the internal pull-up/pull-down device setting; if driven with 0, the driver's pull-down transistor will be activated.

The open drain mode is controlled by the register Px_OD.

Px_OD
Port x Open Drain Control Register

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 7)	n	rw	Port x Pin n Open Drain Mode 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state (default)

Parallel Ports
Pull-Up/Pull-Down Device Register

Internal pull-up/pull-down devices can be optionally applied to a port pin. This offers the possibility to configure the following input characteristics:

- Tristate
- High-impedance with a weak pull-up device
- High-impedance with a weak pull-down device

and the following output characteristics:

- Push/pull (optional pull-up/pull-down)
- Open drain with internal pull-up
- Open drain with external pull-up

The pull-up/pull-down device can be fixed or controlled via the registers Px_PUDSEL and Px_PUDEN. Register Px_PUDSEL selects the type of pull-up/pull-down device, while register Px_PUDEN enables or disables it. The pull-up/pull-down device can be selected pinwise.

Px_PUDSEL
Port x Pull-Up/Pull-Down Select Register

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
rW	rW	rW	rW	rW	rW	rW	rW

Field	Bits	Type	Description
Pn (n = 0 - 7)	n	rW	Pull-Up/Pull-Down Select Port x Bit n 0 Pull-down device is selected 1 Pull-up device is selected

Px_PUDEN
Port x Pull-Up/Pull-Down Enable Register

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
rW	rW	rW	rW	rW	rW	rW	rW

Parallel Ports

Field	Bits	Type	Description
Pn (n = 0 - 7)	n	rw	Pull-Up/Pull-Down Enable at Port x Bit n 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled

Alternate Input Functions

The number of alternate functions that uses a pin for input is not limited. Each port control logic of an I/O pin provides several input paths of digital input value via register. These port input selection registers are described in each peripheral chapter.

Alternate Output Functions

Alternate functions are selected via an output multiplexer which can select up to eight output lines. This multiplexer can be controlled by the following signals:

- Register Px_ALTSEL0
- Register Px_ALTSEL1
- Register Px_ALTSEL2

Selection of alternate functions is defined in registers Px_ALTSEL0, Px_ALTSEL1 and Px_ALTSEL2. Px_ALTSEL2 is optional for ports that does not required more than 4 alternate functions.

Note: Set Px_ALTSEL0.Pn, Px_ALTSEL1.Pn and Px_ALTSEL2.Pn to select only implemented alternate output functions.

Px_ALTSELn (n = 0, 1, 2)

Port x Alternate Select Register

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw

Input Control Register

Input control register controls the input driver of each port pin. This register is use in port P2.

Parallel Ports

Field	Bits	Type	Description
Pn (n = 0 - 7)	n	rw	Pin Output Functions Configuration of Px_ALTSEL0.Pn, Px_ALTSEL1.Pn and Px_ALTSEL2.Pn for GPIO or alternate settings: 000 Normal GPIO 001 Alternate Select 1 010 Alternate Select 2 011 Alternate Select 3 100 Alternate Select 4 101 Alternate Select 5 110 Alternate Select 6 111 Alternate Select 7

Px_EN
Port x Input Control Register

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 7)	n	rw	Input Driver Control at Port x Bit n 0 Input driver is enabled 1 Input driver is disabled

Parallel Ports

11.2 Port 0

Port 0 is a 7-bit general purpose bidirectional port. It uses the standard bidirectional pad for each pin. The registers of Port 0 are summarized in [Table 11-3](#).

Table 11-3 Port 0 Registers

Register Short Name	Register Long Name
P0_DATAIN	Port 0 Data In Register
P0_DATAOUT	Port 0 Data Out Register
P0_OD	Port 0 Open Drain Control Register
P0_PUDSEL	Port 0 Pull-Up/Pull-Down Select Register
P0_PUDEN	Port 0 Pull-Up/Pull-Down Enable Register
P0_ALTSEL0	Port 0 Alternate Select Register 0
P0_ALTSEL1	Port 0 Alternate Select Register 1
P0_ALTSEL2	Port 0 Alternate Select Register 2

11.2.1 Functions

Port 0 input and output functions are shown in [Table 11-4](#).

Table 11-4 Port 0 Input/Output Functions

Port Pin	Input/Output	Select	Connected Signal(s)	From/to Module
P0.0	Input	GPI	P0_DATAIN.P0	
		ALT1	T2_0	Timer 2
		ALT2	T13HR_1	CCU6
		ALT3	MTSR_2	SSC
		ALT4	MRST_3	SSC
		ALT5	T12HR_0	CCU6
		ALT6	CCPOS0_0	CCU6
		ALT7	TSIN0	LEDTSCU
	Output	GPO	P0_DATAOUT.P0	
		ALT1	LINE0/TSIN0	LEDTSCU
		ALT2	MTSR_2	SSC
		ALT3	COU61_1	CCU6

Parallel Ports
Table 11-4 Port 0 Input/Output Functions (cont'd)

Port Pin	Input/Output	Select	Connected Signal(s)	From/to Module
P0.1	Input	GPI	P0_DATAIN.P1	
		ALT1	T0_0	Timer 0
		ALT2	CC61_1	CCU6
		ALT3	MTSR_3	SSC
		ALT4	MRST_2	SSC
		ALT5	T13HR_0	CCU6
		ALT6	CCPOS1_0	CCU6
		ALT7	TSIN1	LEDTSCU
	Output	GPO	P0_DATAOUT.P1	
		ALT1	LINE1/TSIN1	LEDTSCU
		ALT2	MRST_2	SSC
		ALT3	CC61_1	CCU6
P0.2	Input	GPI	P0_DATAIN.P2	
		ALT1	T1_0	Timer 1
		ALT2	CC62_1	CCU6
		ALT3	SCL_1	IIC
		ALT4	–	
		ALT5	–	
		ALT6	CCPOS2_0	CCU6
		ALT7	TSIN2	LEDTSCU
	Output	GPO	P0_DATAOUT.P2	
		ALT1	LINE2/TSIN2	LEDTSCU
		ALT2	SCL_1	IIC
		ALT3	CC62_1	CCU6

Parallel Ports
Table 11-4 Port 0 Input/Output Functions (cont'd)

Port Pin	Input/Output	Select	Connected Signal(s)	From/to Module
P0.3	Input	GPI	P0_DATAIN.P3	
		ALT1	–	
		ALT2	CC60_1	CCU6
		ALT3	SDA_1	IIC
		ALT4	–	
		ALT5	–	
		ALT6	CTR $\overline{\text{AP}}\#_0$	CCU6
		ALT7	TSIN3	LEDTSCU
	Output	GPO	P0_DATAOUT.P3	
		ALT1	LINE3/TSIN3	SSC
		ALT2	SDA_1	IIC
		ALT3	CC60_1	CCU6
P0.4	Input	GPI	P0_DATAIN.P4	
		ALT1	T2EX_1	Timer 2
		ALT2	SCL_0	IIC
		ALT3	SCK_0	SSC
		ALT4	–	
		ALT5	EXINT1_0	SCU
		ALT6	CTR $\overline{\text{AP}}\#_1$	CCU6
		ALT7	TSIN4	LEDTSCU
	Output	GPO	P0_DATAOUT.P4	
		ALT1	LINE4/TSIN4	LEDTSCU
		ALT2	SCK_0	SSC
		ALT3	EXF2_0	Timer 2
		ALT4	SCL_0	IIC
		ALT5	COL0_1	LEDTSCU
		ALT6	COL3_1	LEDTSCU
		ALT7	COLA_2	LEDTSCU

Parallel Ports

Table 11-4 Port 0 Input/Output Functions (cont'd)

Port Pin	Input/Output	Select	Connected Signal(s)	From/to Module
P0.5	Input	GPI	P0_DATAIN.P5	
		ALT1	RXD_0	UART
		ALT2	RTCCLK	RTC
		ALT3	MTSR_0	SSC
		ALT4	MRST_1	SSC
		ALT5	EXINT0_0	SCU
		ALT6	–	
		ALT7	TSIN5	LEDTSCU
	Output	GPO	P0_DATAOUT.P5	
		ALT1	LINE5/TSIN5	LEDTSCU
		ALT2	MTSR_0	SSC
		ALT3	COUT62_1	CCU6
		ALT4	TXD_3	UART
		ALT5	COL1_1	LEDTSCU
		ALT6	EXF2_2	Timer 2

Parallel Ports

Table 11-4 Port 0 Input/Output Functions (cont'd)

Port Pin	Input/Output	Select	Connected Signal(s)	From/to Module
P0.6	Input	GPI	P0_DATAIN.P6	
		ALT1	RXD_1	UART
		ALT2	SDA_0	IIC
		ALT3	MTSR_1	SSC
		ALT4	MRST_0	SSC
		ALT5	EXINT0_1	SCU
		ALT6	T2EX_0	Timer 2
		ALT7	TSIN6	LEDTSCU
		OTHERS	SPD_0	SPD
	Output	GPO	P0_DATAOUT.P6	
		ALT1	LINE6/TSIN6	LEDTSCU
		ALT2	MRST_0	SSC
		ALT3	TXD_0	UART
		ALT4	SDA_0	IIC
		ALT5	COL2_1	LEDTSCU
		ALT6	COLA_1	LEDTSCU
		OTHERS	SPD_0	SPD

11.2.2 Registers Description

P0_DATAIN

Port 0 Data In Register

(86_H)

Reset Value: XX_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	P6	P5	P4	P3	P2	P1	P0
r	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
Pn (n = 0 - 6)	n	rh	Port 0 Pin n Data Value 0 Port 0 pin n data value = 0 1 Port 0 pin n data value = 1
0	7	r	Reserved Returns 0 if read; should be written with 0.

P0_DATAOUT

Port 0 Data Output Register

(80_H)

Reset Value: 7F_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	P6	P5	P4	P3	P2	P1	P0
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 6)	n	rw	Port 0 Pin n Data Value 0 Port 0 pin n data value = 0 1 Port 0 pin n data value = 1
0	7	r	Reserved Returns 0 if read; should be written with 0.

Parallel Ports

P0_OD

Port 0 Open Drain Control Register (80_H)

Reset Value: 7F_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	P6	P5	P4	P3	P2	P1	P0
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 6)	n	rw	Port 0 Pin n Open Drain Mode 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state (default)
0	7	r	Reserved Returns 0 if read; should be written with 0.

P0_PUDSEL

Port 0 Pull-Up/Pull-Down Select Register(80_H)

Reset Value: 6F_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	P6	P5	P4	P3	P2	P1	P0
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 6)	n	rw	Pull-Up/Pull-Down Select Port 0 Bit n 0 Pull-down device is selected 1 Pull-up device is selected (default)
0	7	r	Reserved Returns 0 if read; should be written with 0.

Parallel Ports

P0_PUDEN

Port 0 Pull-Up/Pull-Down Enable Register(86_H)

Reset Value: 50_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	P6	P5	P4	P3	P2	P1	P0
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 6)	n	rw	Pull-Up/Pull-Down Enable at Port 0 Bit n 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	7	r	Reserved Returns 0 if read; should be written with 0.

Note: P0.4 has a default pull down device enabled which will be disabled by firmware in the startup code. At the end of the startup code, it is re-enabled before the user code start to execute.

P0_ALTSEL0

Port 0 Alternate Select Register 0 (80_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0	P6	P5	P4	P3	P2	P1	P0
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 6)	n	rw	See Table 11-5
0	7	r	Reserved Returns 0 if read; should be written with 0.

Parallel Ports

P0_ALTSEL1

Port 0 Alternate Select Register 1 (86_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0	P6	P5	P4	P3	P2	P1	P0
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 6)	n	rw	See Table 11-5
0	7	r	Reserved Returns 0 if read; should be written with 0.

P0_ALTSEL2

Port 0 Alternate Select Register 2 (85_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0	P6	P5	P4			0	
r	rw	rw	rw			r	

Field	Bits	Type	Description
Pn (n = 0 - 6)	n	rw	See Table 11-5
0	[3:0], 7	r	Reserved Returns 0 if read; should be written with 0.

Table 11-5 Function of Bits P0_ALTSEL2.Pn, P0_ALTSEL1.Pn and P0_ALTSEL0.Pn

P0_ALTSEL2.Pn	P0_ALTSEL1.Pn	P0_ALTSEL0.Pn	Function
0	0	0	Normal GPIO (default)
0	0	1	Alternate Select 1
0	1	0	Alternate Select 2

Parallel Ports
Table 11-5 Function of Bits P0_ALTSEL2.Pn, P0_ALTSEL1.Pn (cont'd) and P0_ALTSEL0.Pn

P0_ALTSEL2.Pn	P0_ALTSEL1.Pn	P0_ALTSEL0.Pn	Function
0	1	1	Alternate Select 3
1	0	0	Alternate Select 4
1	0	1	Alternate Select 5
1	1	0	Alternate Select 6
1	1	1	Alternate Select 7

11.3 Port 1

Port 1 is a general purpose bidirectional port. In XC82x, P1 uses the standard bidirectional pad for each pin. The registers of Port 1 are summarized in [Table 11-6](#).

Table 11-6 Port 1 Registers

Register Short Name	Register Long Name
P1_DATAOUT	Port 1 Data Out Register
P1_DATAIN	Port 1 Data In Register
P1_OD	Port 1 Open Drain Control Register
P1_PUDSEL	Port 1 Pull-Up/Pull-Down Select Register
P1_PUDEN	Port 1 Pull-Up/Pull-Down Enable Register
P1_ALTSEL0	Port 1 Alternate Select Register 0
P1_ALTSEL1	Port 1 Alternate Select Register 1

11.3.1 Functions

Port 1 input and output functions are shown in [Table 11-7](#).

Table 11-7 Port 1 Input / Output Functions

Port Pin	Input/Output	Select	Connected Signal(s)	From/to Module
P1.0	Input	GPI	P1_DATAIN.P0	
		ALT 1	RXD_2	UART
		ALT 2	T2EX_2	Timer 2
		ALT 3	–	
		ALT 4	–	
		ALT 5	EXINT0_2	SCU
		ALT 6	–	
		OTHERS	SPD_1	SPD
	Output	GPO	P1_DATAOUT.P0	
		ALT 1	COL0_0	LEDTSCU
		ALT 2	COUT60_0	CCU6
		ALT 3	TXD_1	UART
		OTHERS	SPD_1	SPD
P1.1 ¹⁾	Input	GPI	P1_DATAIN.P1	
		ALT 1	–	
		ALT 2	–	
		ALT 3	–	
		ALT 4	–	
		ALT 5	–	
		ALT 6	CC60_0	CCU6
	Output	GPO	P1_DATAOUT.P1	
		ALT 1	COL1_0	LEDTSCU
		ALT 2	CC60_0	CCU6
		ALT 3	TXD_2	UART

Parallel Ports
Table 11-7 Port 1 Input / Output Functions (cont'd)

Port Pin	Input/Output	Select	Connected Signal(s)	From/to Module
P1.2	Input	GPI	P1_DATAIN.P2	
		ALT 1	–	
		ALT 2	–	
		ALT 3	–	
		ALT 4	–	
		ALT 5	EXINT4	SCU
		ALT 6	–	
	Output	GPO	P1_DATAOUT.P2	
		ALT 1	COL2_0	LEDTSKU
		ALT 2	COUT61_0	CCU6
		ALT 3	COUT63_0	CCU6
P1.3	Input	GPI	P1_DATAIN.P3	
		ALT 1	–	
		ALT 2	–	
		ALT 3	–	
		ALT 4	–	
		ALT 5	–	
		ALT 6	CC61_0	CCU6
	Output	GPO	P1_DATAOUT.P3	
		ALT1	COL3_0	LEDTSKU
		ALT2	CC61_0	CCU6
		ALT3	EXF2_1	Timer 2

Parallel Ports

Table 11-7 Port 1 Input / Output Functions (cont'd)

Port Pin	Input/Output	Select	Connected Signal(s)	From/to Module
P1.4	Input	GPI	P1_DATAIN.P4	
		ALT 1	–	
		ALT 2	–	
		ALT 3	–	
		ALT 4	–	
		ALT 5	EXINT5	SCU
		ALT 6	–	
	Output	GPO	P1_DATAOUT.P4	
		ALT1	COL4	LEDTSCU
		ALT2	COUT62_0	CCU6
		ALT3	COUT63_1	CCU6
P1.5	Input	GPI	P1_DATAIN.P5	
		ALT 1	–	
		ALT 2	–	
		ALT 3	–	
		ALT 4	–	
		ALT 5	–	
		ALT 6	CC62_0	CCU6
	Output	GPO	P1_DATAOUT.P5	
		ALT1	COL5	LEDTSCU
		ALT2	CC62_0	CCU6
		ALT3	COLA_0	LEDTSCU

¹⁾ P1.1, P1.3, P1.4 and P1.5 is only available in XC824

Parallel Ports

11.3.2 Registers Description

P1_DATAIN

Port 1 Data In Register

(91_H)

Reset Value: XX_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	P5	P4	P3	P2	P1	P0	
r	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
Pn (n = 0 - 5)	n	rh	Port 1 Pin n Data Value 0 Port 1 pin n data value = 0 1 Port 1 pin n data value = 1
0	[7:6]	r	Reserved Returns 0 if read; should be written with 0.

P1_DATAOUT

Port 1 Data Out Register

(90_H)

Reset Value: 3F_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	P5	P4	P3	P2	P1	P0	
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 5)	n	rw	Port 1 Pin n Data Value 0 Port 1 pin n data value = 0 1 Port 1 pin n data value = 1
0	[7:6]	r	Reserved Returns 0 if read; should be written with 0.

Parallel Ports

P1_OD

Port 1 Open Drain Control Register (90_H)

Reset Value: 3F_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	P5	P4	P3	P2	P1	P0	
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 5)	n	rw	Port 1 Pin n Open Drain Mode 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state (default)
0	[7:6]	r	Reserved Returns 0 if read; should be written with 0.

P1_PUDSEL

Port 1 Pull-Up/Pull-Down Select Register(90_H)

Reset Value: 3F_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	P5	P4	P3	P2	P1	P0	
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 5)	n	rw	Pull-Up/Pull-Down Select Port 1 Bit n 0 Pull-down device is selected 1 Pull-up device is selected (default)
0	[7:6]	r	Reserved Returns 0 if read; should be written with 0.

Parallel Ports

P1_PUDEN

Port 1 Pull-Up/Pull-Down Enable Register(91_H)

Reset Value: 00_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	P5	P4	P3	P2	P1	P0	
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 5)	n	rw	Pull-Up/Pull-Down Enable at Port 1 Bit n 0 Pull-up or Pull-down device is disabled (default) 1 Pull-up or Pull-down device is enabled
0	[7:6]	r	Reserved Returns 0 if read; should be written with 0.

P1_ALTSELx (x = 0 - 1)

Port 1 Alternate Select Register (90_H + x)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0	P5	P4	P3	P2	P1	P0	
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 5)	n	rw	See Table 11-8
0	[7:6]	r	Reserved Returns 0 if read; should be written with 0.

Table 11-8 Function of Bits P1_ALTSEL0.Pn and P1_ALTSEL1.Pn

P1_ALTSEL1.Pn	P1_ALTSEL0.Pn	Function
0	0	Normal GPIO (default)
0	1	Alternate Select 1

Parallel Ports**Table 11-8 Function of Bits P1_ALTSEL0.Pn and P1_ALTSEL1.Pn (cont'd)**

P1_ALTSEL1.Pn	P1_ALTSEL0.Pn	Function
1	0	Alternate Select 2
1	1	Alternate Select 3

11.4 Port 2

Port 2 is a general purpose input-only port. In XC82x, Port 2 has 4 pins, P2.0 - P2.3. The registers of Port 2 are summarized in [Table 11-9](#).

Table 11-9 Port 2 Registers

Register Short Name	Register Long Name
P2_DATAIN	Port 2 Data Register
P2_EN	Port 2 Enable Register
P2_PUDSEL	Port 2 Pull-Up/Pull-Down Select Register
P2_PUDEN	Port 2 Pull-Up/Pull-Down Enable Register

11.4.1 Functions

Port 2 input and output functions are shown in [Table 11-10](#).

Table 11-10 Port 2 Input Functions

Port Pin	Input/Output	Select	Connected Signal(s)	From/to Module
P2.0	Input	GPI	P2_DATAIN.P0	
		ALT 1	CCPOS0_1	CCU6
		ALT 2	T12HR_2	CCU6
		ALT 3	T13HR_2	CCU6
		ALT 4	T2EX_3	Timer 2
		ALT 5	T2_1	Timer 2
		ALT 6	EXINT0_3	SCU
		ANALOG	AN0	ADC
P2.1	Input	GPI	P2_DATAIN.P1	
		ALT 1	CCPOS1_1	CCU6
		ALT 2	RXD_3	UART
		ALT 3	MTSR_4	SSC
		ALT 4	–	
		ALT 5	T0_1	Timer 0
		ALT 6	EXINT1_1	SCU
		ANALOG	AN1	ADC
P2.2	Input	GPI	P2_DATAIN.P1	
		ALT 1	CCPOS2_1	CCU6
		ALT 2	T12HR_3	CCU6
		ALT 3	T13HR_3	CCU6
		ALT 4	SCK_1	SSC
		ALT 5	T1_1	Timer 1
		ALT 6	EXINT2	SCU
		ANALOG	AN2	ADC

Parallel Ports

Table 11-10 Port 2 Input Functions (cont'd)

Port Pin	Input/Output	Select	Connected Signal(s)	From/to Module
P2.3	Input	GPI	P2_DATAIN.P3	
		ALT 1	CCPOS0_2	CCU6
		ALT 2	CTRAP#_2	CCU6
		ALT 3	–	
		ALT 4	–	
		ALT 5	T2_2	Timer 2
		ALT 6	EXINT3	SCU
		ANALOG	AN3	ADC

11.4.2 Registers Description

P2_DATAIN

Port 2 Data Register

(94_H)

Reset Value: 0X_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0				P3	P2	P1	P0
r				rh	rh	rh	rh

Field	Bits	Type	Description
Pn (n = 0 - 3)	n	rh	Port 2 Pin n Data Value 0 Port 2 pin n data value = 0 1 Port 2 pin n data value = 1
0	[7:4]	r	Reserved Returns 0 if read; should be written with 0.

P2_EN

Port 2 Enable Register

(94_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0				P3	P2	P1	P0
r				rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 3)	n	rw	Port 2 Pin n Input Driver Control 0 Input driver is enabled (default) 1 Input driver is disabled
0	[7:4]	r	Reserved Returns 0 if read; should be written with 0.

Parallel Ports

P2_PUDSEL

Port 2 Pull-Up/Pull-Down Select Register(93_H)

Reset Value: 0F_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0				P3	P2	P1	P0
r				rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 3)	n	rw	Pull-Up/Pull-Down Select Port 2 Bit n 0 Pull-down device is selected 1 Pull-up device is selected (default)
0	[7:4]	r	Reserved Returns 0 if read; should be written with 0.

P2_PUDEN

Port 2 Pull-Up/Pull-Down Enable Register(94_H)

Reset Value: 00_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0				P3	P2	P1	P0
r				rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0 - 3)	n	rw	Pull-Up/Pull-Down Enable at Port 2 Bit n 0 Pull-up or Pull-down device is disabled (default) 1 Pull-up or Pull-down device is enabled
0	[7:4]	r	Reserved Returns 0 if read; should be written with 0.

12 Multiplication/Division Unit

12.1 Overview

The Multiplication/Division Unit (MDU) provides fast 16-bit multiplication, 16-bit and 32-bit division as well as shift and normalize features. It has been integrated to support the XC82x Core in real-time control applications, which require fast mathematical computations.

The MDU uses a total of 14 registers; 12 registers for data manipulation, one register to control the operation of MDU and one register for storing the status flags. These registers are memory mapped as special function registers like any other registers for peripheral control. The MDU operates concurrently with and independent of the CPU.

Features

- Fast signed/unsigned 16-bit multiplication
- Fast signed/unsigned 32-bit divide by 16-bit and 16-bit divide by 16-bit operations
- Signed 16-bit multiplication with single left shift (to support fixed-point number calculations in Q15 format)
- Signed 32-bit divide by 16-bit with single right shift (to support fixed-point number calculations in Q15 format)
- 32-bit unsigned normalize operation
- 32-bit arithmetic/logical shift operations

Multiplication/Division Unit

Table 12-1 specifies the number of clock cycles used for calculation in various operations.

Table 12-1 MDU Operation Characteristics

Operation	Result	Reminder	No. of Clock Cycles Used for Calculation
Signed 32-bit/16-bit	32-bit	16-bit	33
Signed 32-bit/16-bit with Single Right Shift	32-bit	16-bit	33
Signed 16-bit/16bit	16-bit	16-bit	17
Signed 16-bit x 16-bit	32-bit	–	16
Signed 16-bit x 16-bit with Single Left Shift	32-bit	–	16
Unsigned 32-bit/16-bit	32-bit	16-bit	32
Unsigned 16-bit/16-bit	16-bit	16-bit	16
Unsigned 16-bit x 16-bit	32-bit	–	16
32-bit normalize	–	–	No. of shifts + 1 (Max. 32)
32-bit shift L/R	–	–	No. of shifts + 1 (Max. 32)

12.2 System Information

This section provides system information relevant to the MDU.

12.2.1 Clocking Configuration

The MDU runs on the FPCLK at a fixed frequency of 48 MHz.

If the MDU functionality is not required at all, it can be completely disabled by gating off its clock input for maximal power reduction. This is done by setting bit MDU_DIS in register PMCON1 as described below.

The bit field PAGE of SCU_PAGE register must be programmed before accessing the PMCON1 register.

Multiplication/Division Unit

PMCON1

Peripheral Management Control Register 1(EF_H)

Reset Value: DF_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	0	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	r	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MDU_DIS	4	rw	MDU Disable Request. Active high. 0 MDU is in normal operation. 1 Request to disable the MDU. (default)
0	5	r	Reserved Returns 0 if read; should be written with 0.

12.2.2 Interrupt Events and Assignment

Table 12-2 lists the interrupt event sources from the MDU, and the corresponding event interrupt enable bit and flag bit.

Table 12-2 MDU Interrupt Events

Event	Event Interrupt Enable Bit	Event Flag Bit
End-of-Calculation	MDUCON.IE	MDUSTAT.IRDY
Occurrence-of-Error		MDUSTAT.IERR

Table 12-3 shows the interrupt node assignment for each MDU interrupt source.

Table 12-3 MDU Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
End-of-Calculation	IEN1.EX2	—	43 _H
Occurrence-of-Error			

12.3 Functional Description

The MDU can be regarded as a special coprocessor for multiplication, division, normalization and shift. Its operation can be divided into three phases (see **Figure 12-1**).

Multiplication/Division Unit

Phase One: Load MDx Registers

In this phase, the operands are loaded into the MDU Operand (MDx) registers by the CPU.

The type of calculation the MDU must perform is selected by writing a 4-bit opcode that represents the required operation into the bit field MDUCON.OPCODE.

Phase Two: Execute Calculation

This phase commences only when the start bit MDUCON.START is set, which in turn sets the busy flag. The start bit is automatically cleared in the next cycle.

During this phase, the MDU works on its own, in parallel with the CPU. The result of the calculation is made available in the MDU Result (MRx) registers at the end of this phase.

Phase Three: Read Result from the MRx Registers

In this final phase, the result is fetched from the MRx registers by the CPU. The MRx registers will be overwritten at the start of the next calculation phase.

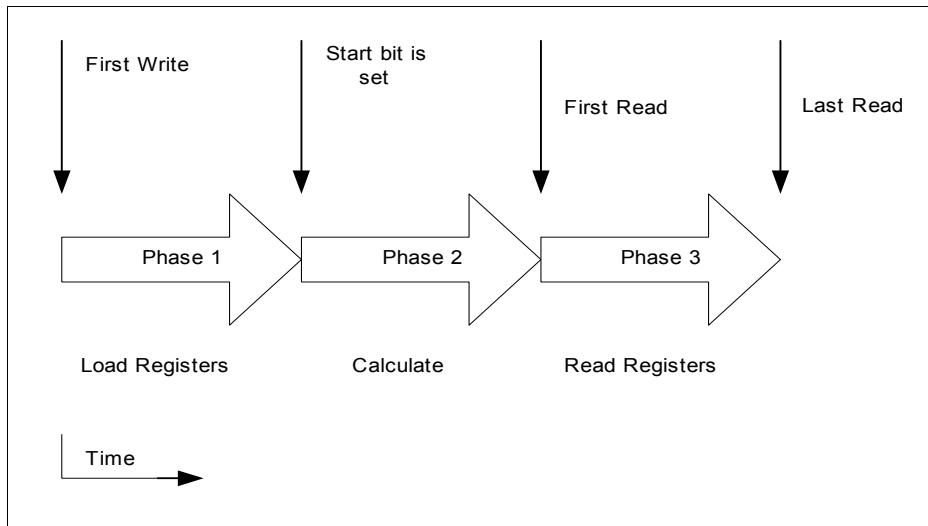


Figure 12-1 Operating Phases of the MDU

12.3.1 Division

The MDU supports the truncated division operation, which is also the ISO C99 standard and the popular choice among modern processors. The division and modulus functions of the truncated division are related in the following way:

Multiplication/Division Unit

If $q = D \div d$
 and $r = D \bmod d$
 then $D = q * d + r$
 and $|r| < |d|$

where “D” is the dividend, “d” is the divisor, “q” is the quotient and “r” is the remainder.

The truncated division rounds the quotient towards zero and the sign of its remainder is always the same as that of its dividend, i.e., $\text{sign}(r) = \text{sign}(D)$.

12.3.2 Normalize

The MDU supports up to 32-bit unsigned normalize.

Normalizing is done on an unsigned 32-bit variable stored in MD0 (least significant byte) to MD3 (most significant byte). This feature is mainly meant to support applications where floating point arithmetic is used. During normalization, all leading zeros of an unsigned 32-bit variable in registers MD0 to MD3 are removed by shift left operations. The whole operation is completed when the MSB (most significant bit) contains a 1.

After normalizing, bit field MR4.SCTR contains the number of shift left operations that were done. This number may be used later as an exponent. The maximum number of shifts in a normalize operation is 31 (= 25 - 1).

12.3.3 Shift

The MDU implements both logical and arithmetic shifts to support up to 32-bit unsigned and signed shift operations.

During logical shift, zeros are shifted in from the left end of register MD3 or right end of register MD0. An arithmetic left shift is identical to a logical left shift, but during arithmetic right shifts, signed bits are shifted in from the left end of register MD3. For example, if the data 0101B and 1010B are to undergo an arithmetic shift right, the results obtained will be 0010B and 1101B, respectively.

For any shift operation, register bit MD4.SLR specifies the shift direction, and MD4.SCTR the shift count.

Note: The MDU does not detect overflows due to an arithmetic shift left operation. User must always ensure that the result of an arithmetic shift left is within the boundaries of MDU.

12.3.4 Multiplication with Single Left Shift

The multiplication with single left shift is similar to a signed 16-bit multiplication except that after the multiplication, the MDU performs a single left shift on the product.

This operation can be used to facilitate signed fixed-point number multiplications in Q15¹⁾ format. Multiplication of two Q15 numbers will result in a Q1.30 fixed-point product. A

Multiplication/Division Unit

single left shift on this product gives a Q31 value and the product in Q15 format can be obtained by taking only the upper 16-bit of the 32-bit value.

12.3.5 Division with Single Right Shift

The division with single right shift is similar to a signed 32-bit by 16-bit division except that the MDU performs a single right shift first before the division.

This operation can be used to facilitate signed 16-bit by 16-bit fixed point number divisions in Q15 format. Normally to divide a Q15 fixed number by another and obtain a Q15 quotient, the dividend has to be scaled up first by a factor of 2^{15} before performing a 32-bit by 16-bit division.

User software can write the 16-bit dividend to the upper half-word of the 32-bit operand (data) register and select the division with single right shift to have the same effect.

12.3.6 Busy Flag

A busy flag is provided to indicate the MDU is still performing a calculation. The flag MDUSTAT.BSY is set at the start of a calculation and cleared after the calculation is completed at the end of phase two. It is also cleared when the error flag is set.

If a second operation needs to be executed, the status of the busy flag will be polled first and only when it is not set, can the start bit be written and the second operation begin. Any unauthorized write to the start bit while the busy flag is still set will be ignored.

12.3.7 Error Detection

The error flag MDUSTAT.IERR is provided to indicate that an error has occurred while performing a calculation. The flag is set by hardware when one of these occurs:

- Division by zero
- Writing of reserved opcodes to MDUCON register

The setting of the error flag causes the current operation to be aborted and triggers an interrupt (see [Section 12.4](#) below). A division by zero error does not set the error flag immediately but rather, at the end of calculation phase for a division operation. An opcode error is detected upon setting MDUCON.START to 1. Errors due to division by zero lead to the loading of a saturated value into the MRx registers.

Note: The accuracy of any result obtained when the error flag is set is not guaranteed by MDU and hence the result should not be used.

1) The Q number is a fixed point number representation that takes the form Qx.y, where x is the number of bits in the integer portion of the number (default is 0) while y is the number of bits in the fractional portion. The signed bit is always excluded from the Qx.y notation.

12.4 Interrupt Generation

The interrupt structure of the MDU is shown in [Figure 12-2](#). There are two possible interrupt events in the MDU, and each event sets one of the two interrupt flags. The interrupt flags are reset by software by writing 0 to it.

At the end of phase two, the interrupt flag MDUSTAT.IRDY is set by hardware to indicate the successful completion of a calculation. The results can then be obtained from the MRx registers. The interrupt line INT_O0 is mapped directly to this interrupt source.

An interrupt can also be triggered when an error occurs during calculation. This is indicated by the setting of the interrupt flag MDUSTAT.IERR. In the event of a division by zero error, MDUSTAT.IERR is set only at the end of the calculation phase. Once the MDUSTAT.IERR is set, any ongoing calculation will be aborted. For division by zero, a saturated value is then loaded into the MRx registers. The bit MDUCON.IR determines the interrupt line to be mapped to this interrupt source.

An interrupt is only generated when interrupt enable bit MDUCON.IE is 1 and the corresponding interrupt event occurs. An interrupt request signal is always asserted positively for 2 CCLK clocks.

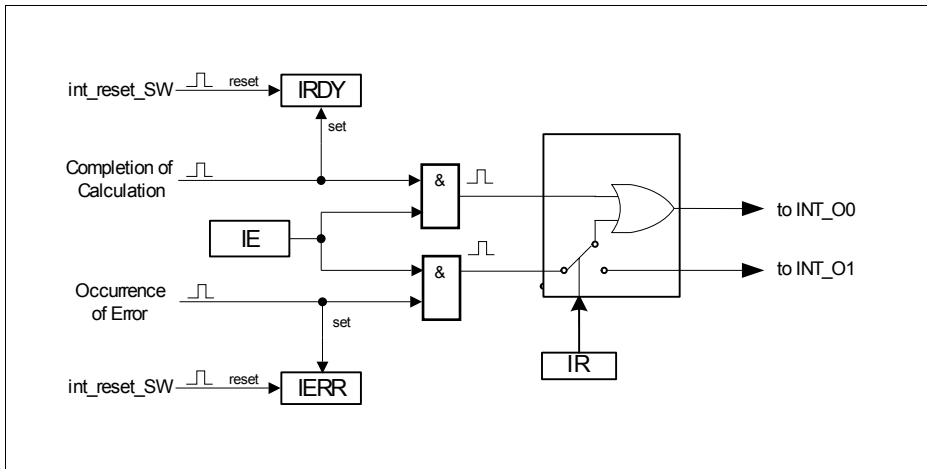


Figure 12-2 Interrupt Generation

Multiplication/Division Unit

12.5 Registers Description

The MDU Special Function Registers are accessed from the standard (non-mapped) SFR area. [Table 12-4](#) lists the MDU registers with their addresses.

Table 12-4 Register Map

SFR	Address	Name
MDU_MDUCON	B1 _H	MDU Control Register
MDU_MDUSTAT	B0 _H	MDU Status Register
MDU_MD0/MR0	B2 _H	MDU Data/Result Register 0
MDU_MD1/MR1	B3 _H	MDU Data/Result Register 1
MDU_MD2/MR2	B4 _H	MDU Data/Result Register 2
MDU_MD3/MR3	B5 _H	MDU Data/Result Register 3
MDU_MD4/MR4	B6 _H	MDU Data/Result Register 4
MDU_MD5/MR5	B7 _H	MDU Data/Result Register 5

The MDx and MRx registers share the same address. However, since MRx registers should never be written to, any write operation to one of these addresses will be interpreted as a write to an MDx register.

In the event of a read operation, an additional bit MDUCON.RSEL is needed to select which set of registers, MDx or MRx, the read operation must be directed to. By default, the MRx registers are read.

The 14 SFRs of the MDU consist of a control register MDUCON, a status register MDUSTAT and 2 sets of data registers, MD0 to MD5 (which contain the operands) and MR0 to MR5 (which contain the results).

Depending on the type of operation, the individual MDx and MRx registers assume specific roles as summarized in [Table 12-5](#) and [Table 12-6](#). For example, in a multiplication operation, the low byte of the 16-bit multiplier must be written to register MD4 and the high byte to MD5.

Table 12-5 MDx Registers

Register	Roles of Registers in Operations			
	16-bit Multiplication	32/16-bit Division	16/16-bit Division	Normalize and Shift
MD0	M'andL	D'endL	D'endL	OperandL
MD1	M'andH	D'end	D'endH	Operand
MD2	—	D'end	—	Operand

Multiplication/Division Unit

Table 12-5 MDx Registers (cont'd)

Register	Roles of Registers in Operations			
	16-bit Multiplication	32/16-bit Division	16/16-bit Division	Normalize and Shift
MD3	–	D'endH	–	OperandH
MD4	M'orL	D'orL	D'orL	Control
MD5	M'orH	D'orH	D'orH	–

Table 12-6 MRx Registers

Register	Roles of Registers in Operations			
	16-bit Multiplication	32/16-bit Division	16/16-bit Division	Normalize and Shift
MR0	PrL	QuoL	QuoL	ResultL
MR1	Pr	Quo	QuoH	Result
MR2	Pr	Quo	–	Result
MR3	PrH	QuoH	–	ResultH
MR4	M'orL	RemL	RemL	Control
MR5	M'orH	RemH	RemH	–

Abbreviations:

- D'end: Dividend, 1st operand of division
- D'or: Divisor, 2nd operand of division
- M'and: Multiplicand, 1st operand of multiplication
- M'or: Multiplier, 2nd operand of multiplication
- Pr: Product, result of multiplication
- Rem: Remainder
- Quo: Quotient, result of division
- ...L: means that this byte is the least significant of the 16-bit or 32-bit operand
- ...H: means that this byte is the most significant of the 16-bit or 32-bit operand

The MDx registers are built with shadow registers, which are latched with data from the actual registers at the start of a calculation. This frees up the MDx registers to be written with the next set of operands while the current calculation is ongoing.

MDx and MRx registers not used in an operation are undefined to the user. For normalize and shift operations, the registers MD4 and MR4 are used as shift input and output control registers to specify the shift direction and store the number of shifts performed.

Multiplication/Division Unit

12.5.1 Operand and Result Registers

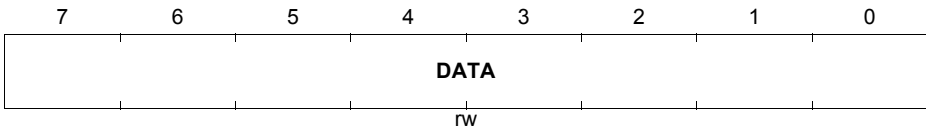
The MDx and MRx registers are used to store the operands and results of a calculation. MD4 and MR4 are also used as input and output control registers for shift and normalize operations.

MDU_MDx (x = 0 - 5)

MDU Data Register x [Operand] ($B2_H + x * 1$)

Reset Value: 00_H

RMAP: 0, **PAGE:** X



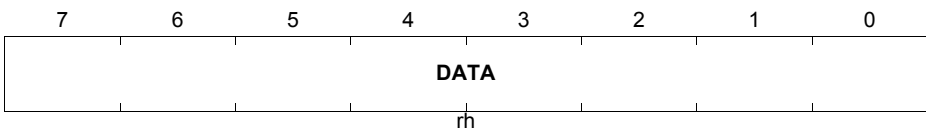
Field	Bits	Type	Description
DATA	[7:0]	rw	Operand Value See Table 12-5 .

MDU_MRx (x = 0 - 5)

MDU Data Register x [Result] ($B2_H + x * 1$)

Reset Value: 00_H

RMAP: 0, **PAGE:** X



Field	Bits	Type	Description
DATA	[7:0]	rh	Result Value See Table 12-6 .

Multiplication/Division Unit

MDU_MD4

Shift Input Control Register

(B6_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
0		SLR	SCTR				
rw		rw	rw				

Field	Bits	Type	Description
SCTR	[4:0]	rw	Shift Counter The count written to SCTR determines the number of shifts to be performed during a shift operation.
SLR	5	rw	Shift Direction 0 _B Selects shift left operation. 1 _B Selects shift right operation.
0	[7:6]	rw	Reserved Should be written with 0. Returns undefined data if read.

MDU_MR4

Shift Output Control Register

(B6_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
0			SCTR				
rh			rh				

Field	Bits	Type	Description
SCTR	[4:0]	rw	Shift Counter After a normalize operation, SCTR contains the number of normalizing shifts performed.
0	[7:5]	rh	Reserved Returns undefined data if read.

Multiplication/Division Unit
12.5.2 Control Register

Register MDUCON contains control bits that select and start the type of operation to be performed.

MDU_MDUCON
MDU Control Register
(B1_H)
Reset Value: 00_H
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
IE	IR	RSEL	START	OPCODE			
rw	rw	rw	rwh	rw			

Field	Bits	Type	Description
OPCODE	[3:0]	rw	Operation Code 0000 _B Unsigned 16-bit Multiplication 0001 _B Unsigned 16-bit/16-bit Division 0010 _B Unsigned 32-bit/16-bit Division 0011 _B 32-bit Logical Shift L/R 0100 _B Signed 16-bit Multiplication 0101 _B Signed 16-bit/16-bit Division 0110 _B Signed 32-bit/16-bit Division 0111 _B 32-bit Arithmetic Shift L/R 1000 _B 32-bit Normalize 1001 _B Signed 16-bit Multiplication with Single Left Shift 1010 _B Signed 32-bit/16-bit Division with Single Right Shift Others: Reserved
START	4	rwh	Start Bit The bit START is set by software and reset by hardware. 0 _B Operation is not started. 1 _B Operation is started.
RSEL	5	rw	Read Select 0 _B Read the MRx registers. 1 _B Read the MDx registers.

Multiplication/Division Unit

Field	Bits	Type	Description
IR	6	rw	Interrupt Routing 0_B The two interrupt sources have their own dedicated interrupt lines. 1_B The two interrupt sources share one interrupt line INT_O0.
IE	7	rw	Interrupt Enable 0_B The interrupt is disabled. 1_B The interrupt is enabled.

Note: Write access to MDUCON is not allowed when the busy flag MDUSTAT.BSY is set during the calculation phase.

Note: Writing reserved opcode values to MDUCON results in an error condition when MDUCON.START bit is set to 1.

Multiplication/Division Unit

12.5.3 Status Register

Register MDUSTAT contains the status flags of the MDU.

MDU_MDUSTAT

MDU Status Register

(B0_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
		0			BSY	IERR	IRDY
		r			rh	rwh	rwh

Field	Bits	Type	Description
IRDY	0	rwh	Interrupt on Result Ready The bit IRDY is set by hardware and reset by software. 0 _B No interrupt is triggered at the end of a successful operation. 1 _B An interrupt is triggered at the end of a successful operation.
IERR	1	rwh	Interrupt on Error The bit IERR is set by hardware and reset by software. 0 _B No interrupt is triggered with the occurrence of an error. 1 _B An interrupt is triggered with the occurrence of an error.
BSY	2	rh	Busy Bit 0 _B The MDU is not running any calculation. 1 _B The MDU is still running a calculation.
0	[7:3]	r	Reserved Returns 0 if read; should be written with 0.

13 Timer 0 and Timer 1

13.1 Overview

Timer 0 and Timer 1 can function as both timers or counters. When functioning as a timer, Timer 0 and Timer 1 are incremented every machine cycle, i.e. every 2 input clocks (or 2 PCLKs). When functioning as a counter, Timer 0 and Timer 1 are incremented in response to a 1-to-0 transition (falling edge) at their respective external input pins, T0 or T1.

They are useful in many timing applications such as measuring the time interval between events, counting events and generating signals at regular intervals. In particular, Timer 1 can be used as the baud-rate generator for the on-chip serial port.

Features:

- Four operational modes :
 - Mode 0: 13-bit timer/counter
 - Mode 1: 16-bit timer/counter
 - Mode 2: 8-bit timer/counter with auto-reload
 - Mode 3: Two 8-bit timers/counters

13.2 System Information

This section provides system information relevant to the Timer 0 and 1.

13.2.1 Pinning

Timer 0 and 1 can be used as an event counter which count 1-to-0 transitions at their external input pins, T0 and T1. These pins are selected from two different sources, T0_0 and T0_1 for Timer 0, and T1_0 and T1_1 for Timer 1. The Timer 0 and 1 pin assignment for XC82x is shown in [Table 13-1](#). This selection is performed by the SFR bits MODPISEL2.T0IS and MODPISEL2.T1IS.

Table 13-1 Timer 0 and 1 Pin Functions in XC82x

Pin	Function	Description	Selected By
P0.1	T0_0	Timer 0 Input	MODPISEL2.T0IS = 0 _B
P2.1	T0_1		MODPISEL2.T0IS = 1 _B
P0.2	T1_0	Timer 1 Input	MODPISEL2.T1IS = 0 _B
P2.2	T1_0		MODPISEL2.T1IS = 0 _B

Timer 0 and Timer 1

The bit field PAGE of SCU_PAGE register must be programmed before accessing the MODPSEL2 register.

MODPSEL2

Peripheral Input Select Register 2 (F5_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	T0IS	T1IS	T2IS				
r	rw	rw	rw			rw	

Field	Bits	Type	Description
T1IS	5	rw	Timer 1 Input Select 0 Timer 1 Input T1_0 is selected. 1 Timer 1 Input T1_1 is selected.
T0IS	6	rw	Timer 0 Input Select 0 Timer 0 Input T0_0 is selected. 1 Timer 0 Input T0_1 is selected.
0	7	r	Reserved Returns 0 if read; should be written with 0.

13.2.2 Clocking Configuration

The Timer 0 and 1 runs on the PCLK at a frequency of either 8 MHz or 24 MHz.

13.2.3 Interrupt Events and Assignment

Table 13-2 lists the interrupt event sources from the Timer 0 and 1, and the interrupt node assignment for each Timer 0 and 1 interrupt source.

Table 13-2 Timer 0 and 1 Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
Timer 0 Overflow	IEN0.ET0	TCON.TF0	0B _H
Timer 1 Overflow	IEN0.ET1	TCON.TF1	1B _H

13.3 Basic Timer Operations

The operations of the two timers are controlled using the Special Function Registers (SFRs) TCON and TMOD. To enable a timer, i.e., allow the timer to run, its control bit TCON.TRx is set. To select the timer input to be either from internal system clock or external pin, the input selector bit TMOD is used.

Note: The "x" (e.g., TCON.TRx) in this chapter denotes either 0 or 1.

Each timer consists of two 8-bit registers - TLx (low byte) and THx (high byte) which defaults to 00_H on reset. Setting or clearing TCON.TRx does not affect the timer registers.

Timer Overflow

When a timer overflow occurs, the timer overflow flag, TCON.TFx, is set, and an interrupt may be raised if the interrupt enable control bit, IEN0.ETx, is set. The overflow flag is automatically cleared when the interrupt service routine is entered.

When Timer 0 operates in mode 3, the Timer 1 control bits, TR1, TF1 and ET1 are reserved for TH0, see [Section 13.4.4](#).

External Control

In addition to pure software control, the timers can also be enabled or disabled through external port control. When external port control is used, SFR EXICON0 must first be configured to bypass the edge detection circuitry for EXINTx to allow direct feed-through. When the timer is enabled (TCON.TRx = 1) and TMOD.GATEx is set, the respective timer will only run if the core external interrupt EXINTx = 1. This facilitates pulse width measurements. However, this is not applicable for Timer 1 in mode 3.

If TMOD.GATEx is cleared, the timer reverts to pure software control.

Timer 0 and Timer 1

13.4 Timer Modes

Timers 0 and 1 are fully compatible and can be configured in four different operating modes, as shown in [Table 13-3](#). The bit field TxM in register TMOD selects the operating mode to be used for each timer.

In modes 0, 1 and 2, the two timers operate independently, but in mode 3, their functions are specialized.

Table 13-3 Timer 0 and Timer 1 Modes

Mode	Operation
0	13-bit timer/counter The timer is essentially an 8-bit counter with a divide-by-32 prescaler. This mode is included solely for compatibility with Intel 8048 devices.
1	16-bit timer/counter The timer registers, TLx and THx, are concatenated to form a 16-bit timer/counter.
2	8-bit timer/counter with auto-reload The timer register TLx is reloaded with a user-defined 8-bit value in THx upon overflow.
3	Timer 0 operates as two 8-bit timers/counters The timer registers, TL0 and TH0, operate as two separate 8-bit counters. Timer 1 is halted and retains its count even if enabled.

Timer 0 and Timer 1

13.4.2 Mode 1

Mode 1 operation is similar to that of mode 0, except that the timer register runs with all 16 bits. Mode 1 operation for Timer 0 is shown in [Figure 13-2](#).

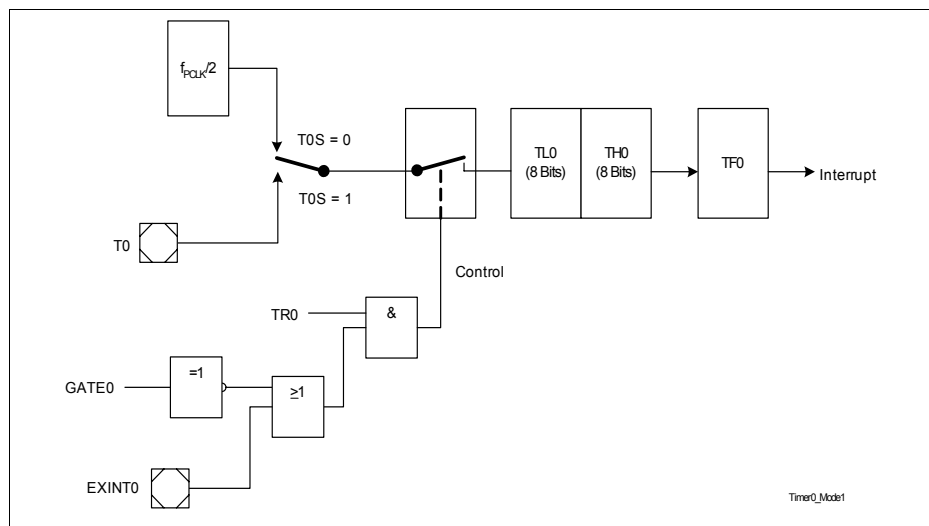


Figure 13-2 Timer 0, Mode 1: 16-Bit Timer/Counter

Timer 0 and Timer 1

13.4.3 Mode 2

In Mode 2 operation, the timer is configured as an 8-bit counter (TLx) with automatic reload, as shown in [Figure 13-3](#) for Timer 0.

An overflow from TLx not only sets TFx, but also reloads TLx with the contents of THx that has been preset by software. The reload leaves THx unchanged.

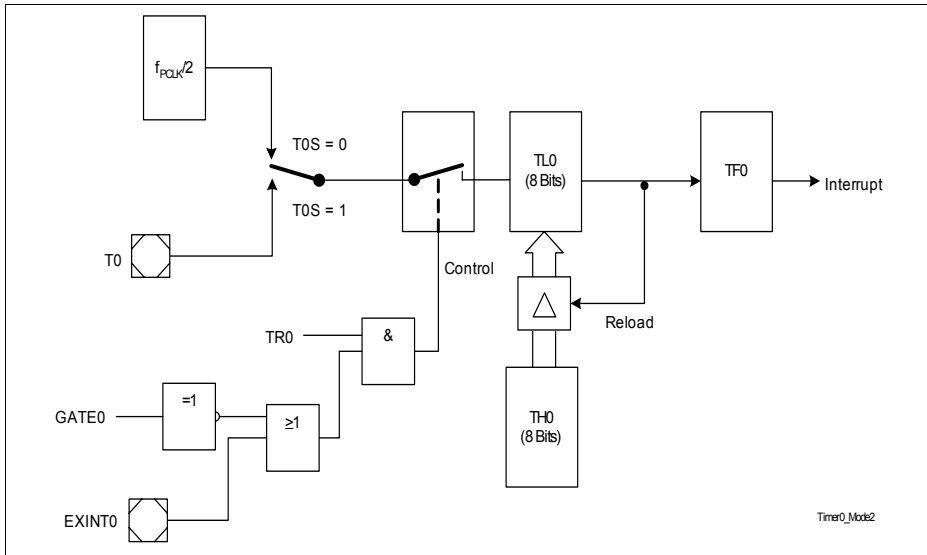


Figure 13-3 Timer 0, Mode 2: 8-Bit Timer/Counter with Auto-Reload

Timer 0 and Timer 1

13.4.4 Mode 3

In mode 3, Timer 0 and Timer 1 behave differently. Timer 0 in mode 3 establishes TL0 and TH0 as two separate counters. Timer 1 in mode 3 simply holds its count. The effect is the same as setting $TR1 = 0$

The logic for mode 3 operation for Timer 0 is shown in [Figure 13-4](#). TL0 uses the Timer 0 control bits GATE0, TR0 and TF0, while TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus, TH0 now sets TF1 upon overflow and generates an interrupt if ET1 is set.

Mode 3 is provided for applications requiring an extra 8-bit timer. When Timer 0 is in mode 3 and TR1 is set, Timer 1 can be turned on by switching it to any of the other modes and turned off by switching it into mode 3.

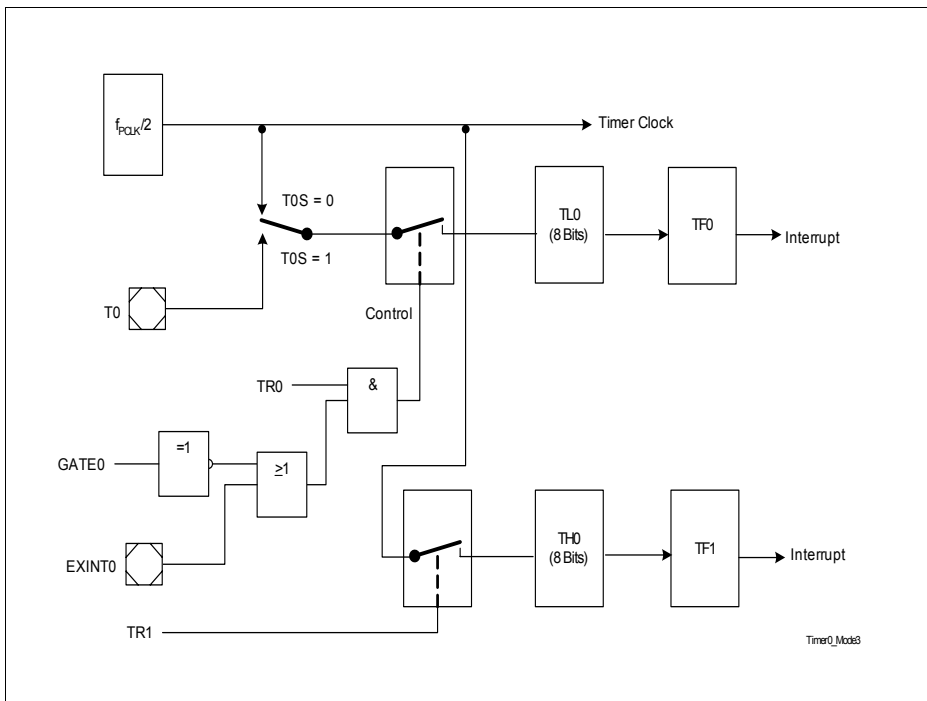


Figure 13-4 Timer 0, Mode 3: Two 8-Bit Timers/Counters

13.5 Registers Description

Seven SFRs control the operations of Timer 0 and Timer 1. They can be accessed from both the standard (non-mapped) and mapped SFR area.

Table 13-4 lists the addresses of these SFRs.

Table 13-4 Register Map

Address	Register
88 _H	TCON
89 _H	TMOD
8A _H	TL0
8B _H	TL1
8C _H	TH0
8D _H	TH1
A8 _H	IEN0

13.5.1 Timer 0 and Timer 1 Registers

The low bytes(TL0, TL1) and high bytes(TH0, TH1)of both Timer 0 and Timer 1 can be combined to a one-timer configuration depending on the mode used. Register TCON controls the operations of Timer 0 and Timer 1. The operating modes of both timers are selected using register TMOD. Register IEN0 contains bits that enable interrupt operations in Timer 0 and Timer 1.

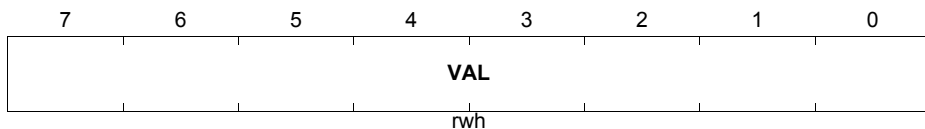
TLx (x = 0 - 1)

Timer x, Low Byte

(8A_H + x)

Reset Value: 00_H

RMAP: X, PAGE: X



Timer 0 and Timer 1

Field	Bits	Type	Description
VAL	[7:0]	rwh	Timer 0/1 Low Register 00 _B TLx holds the 5-bit prescaler value. 01 _B TLx holds the lower 8-bit part of the 16-bit timer value. 10 _B TLx holds the 8-bit timer value. 11 _B TL0 holds the 8-bit timer value; TL1 is not used.

THx (x = 0 - 1)
Timer x, High Byte
(8C_H + x)
Reset Value: 00_H
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
VAL							
rwh							

Field	Bits	Type	Description
VAL	[7:0]	rwh	Timer 0/1 High Register 00 _B THx holds the 8-bit timer value. 01 _B THx holds the higher 8-bit part of the 16-bit timer value. 10 _B THx holds the 8-bit reload value. 11 _B TH0 holds the 8-bit timer value; TH1 is not used.

TCON
Timer 0/1 Control Registers
(88_H)
Reset Value: 00_H
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
rwh	rw	rwh	rw	rwh	rw	rwh	rw

Timer 0 and Timer 1

Field	Bits	Type	Description
TR0	4	rw	Timer 0 Run Control 0 _B Timer is halted 1 _B Timer runs
TF0	5	rwh	Timer 0 Overflow Flag Set by hardware when Timer 0 overflows. Cleared by hardware when the processor calls the interrupt service routine.
TR1	6	rw	Timer 1 Run Control 0 _B Timer is halted 1 _B Timer runs <i>Note: Timer 1 Run Control affects TH0 also if Timer 0 operates in Mode 3</i>
TF1	7	rwh	Timer 1 Overflow Flag Set by hardware when Timer 1 ¹⁾ overflows. Cleared by hardware when the processor calls the interrupt service routine.

1) TF1 is set by TH0 instead if Timer 0 operates in Mode 3.

TMOD

Timer Mode Register

(89_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
GATE1	T1S	T1M	GATE0	T0S	T0M		
rw	rw	rw	rw	rw	rw		

Timer 0 and Timer 1

Field	Bits	Type	Description
T0M	[1:0]	rw	Mode Select Bits 00 _B 13-bit timer (M8048 compatible mode) 01 _B 16-bit timer 10 _B 8-bit auto-reload timer 11 _B Timer 0 is split into two halves. TL0 is an 8-bit timer controlled by the standard Timer 0 control bits, and TH0 is the other 8-bit timer controlled by the standard Timer 1 control bits. TH1 and TL1 of Timer 1 are held (Timer 1 is stopped).
T0S	2	rw	Timer 0 Selector 0 _B Input is from internal system clock 1 _B Input is from T0 pin
GATE0	3	rw	Timer 0 Gate Flag 0 _B Timer 0 will only run if TCON.TR0 = 1 (software control) 1 _B Timer 0 will only run if EXINT0 pin = 1 (hardware control) and TCON.TR0 is set
T1M	[5:4]	rw	Mode Select Bits 00 _B 13-bit timer (M8048 compatible mode) 01 _B 16-bit timer 10 _B 8-bit auto-reload timer 11 _B Timer 0 is split into two halves. TL0 is an 8-bit timer controlled by the standard Timer 0 control bits, and TH0 is the other 8-bit timer controlled by the standard Timer 1 control bits. TH1 and TL1 of Timer 1 are held (Timer 1 is stopped).
T1S	6	rw	Timer 1 Selector 0 _B Input is from internal system clock 1 _B Input is from T1 pin
GATE1	7	rw	Timer 1 Gate Flag 0 _B Timer 1 will only run if TCON.TR1 = 1 (software control) 1 _B Timer 1 will only run if EXINT0 pin = 1 (hardware control) and TCON.TR1 is set

Timer 0 and Timer 1

IEN0

Interrupt Enable Register

(A8_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
EA	0	ET2	ES	ET1	EX1	ET0	EX0
rw	r	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ET0	1	rw	Timer 0 Overflow Interrupt Enable 0 _B Timer 0 interrupt is disabled 1 _B Timer 0 interrupt is enabled
ET1	3	rw	Timer 1 Overflow Interrupt Enable 0 _B Timer 1 interrupt is disabled 1 _B Timer 1 interrupt is enabled <i>Note: When Timer 0 operates in Mode 3, this interrupt indicates an overflow in the Timer 0 register, TH0.</i>

14 Timer 2

14.1 Overview

Timer 2 is a 16-bit general purpose timer which is functionally compatible with the Timer 2 in the C501 product family. Timer 2 can function as a timer or counter in each of its modes. As a timer, it counts with an input clock of PCLK/12 (if prescaler is disabled). As a counter, Timer 2 counts 1-to-0 transitions on pin T2. In the counter mode, the maximum resolution for the count is PCLK/24 (if prescaler is disabled).

Features

- 16-bit auto-reload mode
- selectable up or down counting
- One channel, 16-bit capture mode

14.2 System Information

This section provides system information relevant to the Timer 2.

14.2.1 Pinning

Timer 2 can be used as an event counter which count 1-to-0 transitions at the external input pin, T2. These pins are selected from four different sources, T2_0, T2_1, T2_2 and T2_3. This selection is performed by the SFR bits MODPSEL2.T2IS. In addition, there are eight sources of T2EX pin for Timer 2. They can be selected by MODPSEL2.T2EXIS. Among these sources, 4 of them are available as external pin. 2 of them are triggered internally by Out of Range 0 (ORC0) event and Out of Range 1 (ORC1) event from the ADC module.

The Timer 2 pin assignment for XC82x is shown in [Table 14-1](#).

Table 14-1 Timer 2 Pin Functions in XC82x

Pin/Sources	Function	Description	Selected By
P0.0	T2_0	Timer 2 Input	MODPSEL2.T2IS = 00 _B
P2.0	T2_1		MODPSEL2.T2IS = 01 _B
P2.3	T2_2		MODPSEL2.T2IS = 10 _B

Table 14-1 Timer 2 Pin Functions in XC82x

Pin/Sources	Function	Description	Selected By
P0.6	T2EX_0	Timer 2 External Trigger Input	MODPISEL2.T2EXIS = 000 _B
P0.4	T2EX_1		MODPISEL2.T2EXIS = 001 _B
P1.0	T2EX_2		MODPISEL2.T2EXIS = 010 _B
P2.0	T2EX_3		MODPISEL2.T2EXIS = 011 _B
ORC0 event	T2EX_4		MODPISEL2.T2EXIS = 100 _B
ORC1 event	T2EX_5		MODPISEL2.T2EXIS = 101 _B

The bit field PAGE of SCU_PAGE register must be programmed before accessing the MODPISEL2 register.

MODPISEL2
Peripheral Input Select Register 2 (F5_H)
Reset Value: 00_H
RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	T0IS	T1IS	T2IS	T2EXIS			
r	rw	rw	rw	rw			

Field	Bits	Type	Description
T2EXIS	[2:0]	rw	Timer 2 External Input Select 000 Timer 2 Input T2EX_0 is selected. 001 Timer 2 Input T2EX_1 is selected. 010 Timer 2 Input T2EX_2 is selected. 011 Timer 2 Input T2EX_3 is selected. 100 Timer 2 Input T2EX_4 is selected. 101 Timer 2 Input T2EX_5 is selected. 110 Reserved. 111 Reserved. <i>Note: T2EX_4 and T2EX_5 are triggered by Out of Range 0 event and Out of Range 1 event respectively.</i>

Timer 2

Field	Bits	Type	Description
T2IS	[4:3]	rw	Timer 2 Input Select 00 Timer 2 Input T2_0 is selected. 01 Timer 2 Input T2_1 is selected. 10 Timer 2 Input T2_2 is selected. 11 Reserved.
0	7	r	Reserved Returns 0 if read; should be written with 0.

14.2.2 Clocking Configuration

The Timer 2 runs on the PCLK at a frequency of either 8 MHz or 24 MHz.

If the Timer 2 functionality is not required at all, it can be completely disabled by gating off its clock input for maximal power reduction. This is done by setting bit T2_DIS in register PMCON1 as described below.

PMCON1

Peripheral Management Control Register 1(EF_H)

Reset Value: FF_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	0	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	r	rw	rw	rw	rw	rw

Field	Bits	Type	Description
T2_DIS	3	rw	T2 Disable Request. Active high. 0 T2 is in normal operation. 1 Request to disable the T2. (default)
0	5	r	Reserved Returns 0 if read; should be written with 0.

14.2.3 Interrupt Events and Assignment

Table 14-2 lists the interrupt event sources from the Timer 2, and the corresponding event interrupt enable bit and flag bit.

Table 14-2 Timer 2 Interrupt Events

Event	Event Interrupt Enable Bit	Event Flag Bit
Timer 2 Overflow	T2_T2CON1.TF2EN	T2_T2CON.TF2
Timer 2 External	T2_T2CON1.EXF2EN	T2_T2CON.EXF2

Table 14-3 shows the interrupt node assignment for each Timer 2 interrupt source.

Table 14-3 Timer 2 Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
Timer 2 Overflow	IEN0.ET2	—	2B _H
Timer 2 External			

14.2.4 IP interconnection

In XC82x, T2EX_4 input and T2EX_5 input can be triggered by Out of Range 0 (ORC0) event and Out of Range 1 (ORC1) event from the ADC module respectively as shown in **Table 14-4**. ORCx will generate a rising edge signal to the T2EX input when there is an out of range event. To detect a rising edge signal in T2EX input, either bit EDGESEL or T2REGS of T2_T2MODE register must be set to 1 depending on the usage of the pin. In addition, T2EX_4 or T2EX_5 input needs to be setup via MODPISEL2 register as shown in **Table 14-1**.

Table 14-4 Timer 2 Interconnections

Connected Other Module Function/Signal	Timer 2 Function/Signal
Out of Range 0 (o): ORC0	Timer 2 External Trigger (i): T2EX_4
Out of Range 1 (o): ORC1	Timer 2 External Trigger (i): T2EX_5

14.2.5 Module Suspend Control

When the On-Chip Debug Support (OCDS) is in Monitor Mode (MMCR2.MMODE = 1) and the Debug-Suspend signal is active (MMCR2.DSUSP = 1), the timer/counter in Timer 2 module in XC82x can be suspended based on the settings of their corresponding module suspend bits in register MODSUSP. When suspended, only the timer stops counting as the counter input clock is gated off. The module is still clocked so that module registers are accessible. Refer to **Chapter 10.2.4** for the definition of register MODSUSP.

14.3 Basic Timer Operations

Timer 2 can be started by using TR2 bit by hardware or software. Timer 2 can be started by setting TR2 bit by software. If bit T2RHEN is set, Timer 2 can be started by hardware. Bit T2REGS defines the event on pin T2EX, falling edge or rising edge, that can set the run bit TR2 by hardware. Timer 2 can only be stopped by resetting TR2 bit by software.

14.4 Auto-Reload Mode

The auto-reload mode is selected when the bit $\overline{\text{CP/RL2}}$ in register T2CON is zero. In this mode, Timer 2 counts to an overflow value and then reloads its register contents with a 16-bit start value for a fresh counting sequence. The overflow condition is indicated by setting bit TF2 in the T2CON register. At the same time, an interrupt request to the core will be generated (if interrupt is enabled). The overflow flag TF2 must be cleared by software.

The auto-reload mode is further classified into two categories depending upon the DCEN control bit in register T2MOD.

14.4.1 Up/Down Count Disabled

If DCEN = 0, the up-down count selection is disabled. The timer, therefore, functions as a pure up counting timer only. The operational block diagram is shown in [Figure 14-1](#).

If the T2CON register bit EXEN2 = 0, the timer starts to count up to a maximum of FFFF_{H} once the timer is started by setting the bit TR2 in register T2CON to 1. Upon overflow, bit TF2 is set and the timer register is reloaded with the 16-bit reload value of the RC2 register. This reload value is chosen by software, prior to the occurrence of an overflow condition. A fresh count sequence is started and the timer counts up from this reload value as in the previous count sequence.

If EXEN2 = 1, the timer counts up to a maximum of FFFF_{H} once TR2 is set. A 16-bit reload of the timer registers from register RC2 is triggered either by an overflow condition or by a negative/positive edge (chosen by the bit EDGESEL in register T2MOD) at input pin T2EX. If an overflow caused the reload, the overflow flag TF2 is set. If a negative/positive transition at pin T2EX caused the reload, bit EXF2 in register T2CON is set. In either case, an interrupt is generated to the core if the related interrupt enable bit EXF2EN/TF2EN in register T2CON1 is enabled and the timer proceeds to its next count sequence. The EXF2 flag, similar to the TF2, must be cleared by software.

If bit T2RHEN is set, Timer 2 is started by first falling edge/rising edge at pin T2EX, which is defined by bit T2REGS. If bit EXEN2 is set, bit EXF2 is also set at the same point when Timer 2 is started with the same falling edge/rising edge at pin T2EX, which is defined by bit EDGESEL. The reload will happen with the following negative/positive transitions at pin T2EX, which is defined by bit EDGESEL.

Timer 2

underflow condition sets the TF2 flag and causes $FFFF_H$ to be reloaded into the THL2 register. A fresh down counting sequence is started and the timer counts down as in the previous counting sequence.

If bit T2RHEN is set, Timer 2 can only be started either by rising edge (T2REGS = 1) at pin T2EX and then proceed with the up counting, or be started by falling edge (T2REGS = 0) at pin T2EX and then proceed with the down counting.

In this mode, bit EXF2 toggles whenever an overflow or an underflow condition is detected. This flag, however, does not generate an interrupt request.

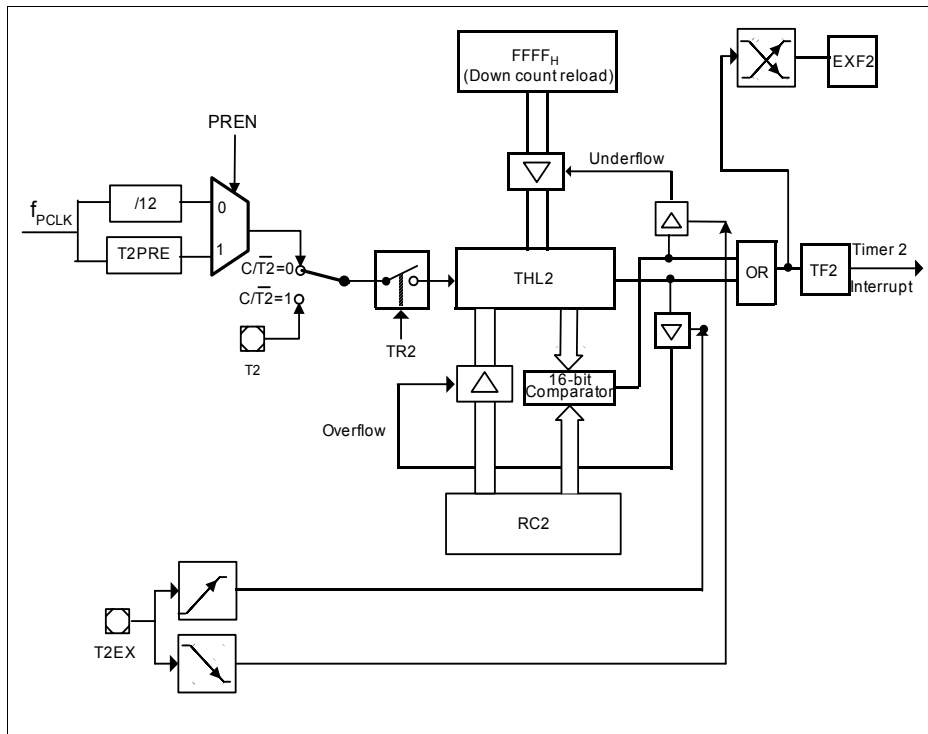


Figure 14-2 Auto-Reload Mode (DCEN = 1)

14.5 Capture Mode

In order to enter the 16-bit capture mode, bits $CP/\overline{RL2}$ and EXEN2 in register T2CON must be set. In this mode, the down count function must remain disabled. The timer functions as a 16-bit timer and always counts up to $FFFF_H$, after which, an overflow condition occurs. Upon overflow, bit TF2 is set and the timer reloads its registers with 0000_H . The setting of TF2 generates an interrupt request to the core.

Additionally, with a falling/rising edge (chosen by T2MOD.EDGESEL) on pin T2EX, the contents of the timer register (THL2) are captured into the RC2 register. The external input is sampled in every PCLK clock cycle. When a sampled input shows a low (high) level in one PCLK clock cycle and a high (low) in the next PCLK clock cycle, a transition is recognized. If the capture signal is detected while the counter is being incremented, the counter is first incremented before the capture operation is performed. This ensures that the latest value of the timer register is always captured.

If bit T2RHEN is set, Timer 2 is started by first falling edge/rising edge at pin T2EX, which is defined by bit T2REGS. If bit EXEN2 is set, bit EXF2 is also set at the same point when Timer 2 is started with the same falling edge/rising edge at pin T2EX, which is defined by bit EDGESEL. The capture will happen with the following negative/positive transitions at pin T2EX, which is defined by bit EDGESEL.

When the capture operation is completed, bit EXF2 is set and can be used to generate an interrupt request. [Figure 14-3](#) describes the capture function of Timer 2.

14.7 External Interrupt Function

While the timer/counter function is disabled ($TR2 = 0$), it is still possible to generate a Timer 2 interrupt to the core via an external event at T2EX, as long as Timer 2 remains enabled ($PMCON1.T2_DIS = 0$). To achieve this, bit EXEN2 in register T2CON must be set. As a result, any transition on T2EX will cause either a dummy reload or a dummy capture, depending on the CP/RL2 bit selection.

By disabling the timer/counter function, T2EX can be alternatively used to provide an edge-triggered (rising or falling) external interrupt function, with bit EXF2 serving as the external interrupt flag.

14.8 Registers Description

All Timer 2 register names described in the following sections are referenced in other chapters of this document with the module name prefix "T2_", e.g., T2_T2CON.

The Timer 2 SFRs are located in the standard (non-mapped) SFR area. [Table 14-5](#) lists these addresses.

Table 14-5 Register Map

Address	Register
C0 _H	T2CON
C1 _H	T2MOD
C2 _H	RC2L
C3 _H	RC2H
C4 _H	T2L
C5 _H	T2H
C6 _H	T2CON1

14.8.1 Mode Register

The T2MOD is used to configure Timer 2 for various modes of operation.

T2_T2MOD

Timer 2 Mode Register

(C1_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
T2REGS	T2RHEN	EDGESEL	PREN	T2PRE		DCEN	
rw	rw	rw	rw	rw		rw	

Field	Bit	Type	Description
DCEN	0	rw	Up/Down Counter Enable 0 _B Up/Down Counter function is disabled 1 _B Up/Down Counter function is enabled and controlled by pin T2EX (Up = 1, Down = 0)
T2PRE	[3:1]	rw	Timer 2 Prescaler Bit Selects the input clock for Timer 2 which is derived from the peripheral clock. 000 _B $f_{T2} = f_{PCLK}$ 001 _B $f_{T2} = f_{PCLK} / 2$ 010 _B $f_{T2} = f_{PCLK} / 4$ 011 _B $f_{T2} = f_{PCLK} / 8$ 100 _B $f_{T2} = f_{PCLK} / 16$ 101 _B $f_{T2} = f_{PCLK} / 32$ 110 _B $f_{T2} = f_{PCLK} / 64$ 111 _B $f_{T2} = f_{PCLK} / 128$
PREN	4	rw	Prescaler Enable 0 _B Prescaler is disabled and the 2 or 12 divider takes effect. 1 _B Prescaler is enabled (see T2PRE bit) and the 2 or 12 divider is bypassed.
EDGESEL	5	rw	Edge Select in Capture Mode/Reload Mode 0 _B The falling edge at Pin T2EX is selected. 1 _B The rising edge at Pin T2EX is selected.
T2RHEN	6	rw	Timer 2 External Start Enable 0 _B Timer 2 External Start is disabled. 1 _B Timer 2 External Start is enabled.

Timer 2

Field	Bit	Type	Description
T2REGS	7	rw	Edge Select for Timer 2 External Start
			<div> <div>0_B</div> <div>The falling edge at Pin T2EX is selected.</div> </div> <div> <div>1_B</div> <div>The rising edge at Pin T2EX is selected.</div> </div>

14.8.2 Control Register

Control register T2CON is used to control the operating modes and interrupt of Timer 2. In addition, it contains the status flags for interrupt generation.

T2_T2CON

Timer 2 Control Register

(C0_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
TF2	EXF2	0	0	EXEN2	TR2	C_T2	CP_RL2
rw	rw	r	r	rw	rw	rw	rw

Field	Bit	Type	Description
CP_RL2	0	rw	Capture/Reload Select
			<div> <div>0_B</div> <div>Reload upon overflow or upon negative/positive transition at pin T2EX (when EXEN2 = 1).</div> </div> <div> <div>1_B</div> <div>Capture Timer 2 data register contents on the negative/positive transition at pin T2EX, provided EXEN2 = 1. The negative or positive transition at Pin T2EX is selected by bit EDGESEL.</div> </div>
C_T2	1	rw	Timer or Counter Select <div> <div>0_B</div> <div>Timer function selected.</div> </div> <div> <div>1_B</div> <div>Count upon negative edge at pin T2.</div> </div>
TR2	2	rw	Timer 2 Start/Stop Control
			<div> <div>0_B</div> <div>Stop Timer 2.</div> </div> <div> <div>1_B</div> <div>Start Timer 2.</div> </div>
EXEN2	3	rw	Timer 2 External Enable Control
			<div> <div>0_B</div> <div>External events are disabled.</div> </div> <div> <div>1_B</div> <div>External events are enabled in Capture/Reload Mode.</div> </div>

Timer 2

Field	Bit	Type	Description
EXF2	6	rwh	Timer 2 External Flag In Capture/Reload Mode, this bit is set by hardware when a negative/positive transition occurs at pin T2EX, if bit EXEN2 = 1. This bit must be cleared by software. <i>Note: When bit DCEN = 1 in auto-reload mode, no interrupt request to the core is generated.</i>
TF2	7	rwh	Timer 2 Overflow/Underflow Flag Set by a Timer 2 overflow/underflow. Must be cleared by software.
0	[5:4]	r	Reserved Returns 0 if read; must be written with 0.

Register T2CON1 is used to enable the external interrupt and the overflow interrupt.

T2_T2CON1

Timer 2 Control Register 1

(C6_H)

Reset Value: 03_H

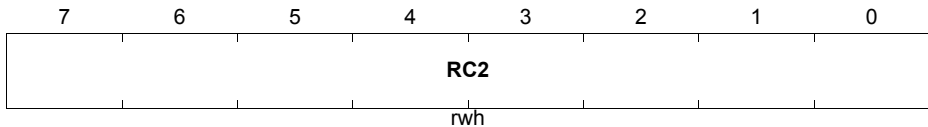
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
0						TF2EN	EXF2EN
r						rw	rw

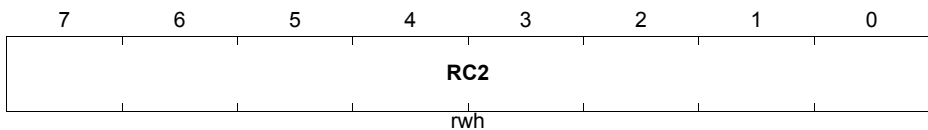
Field	Bit	Type	Description
EXF2EN	0	rw	External Interrupt Enable 0 _B External interrupt is disabled. 1 _B External interrupt is enabled.
TF2EN	1	rw	Overflow/Underflow Interrupt Enable 0 _B Overflow/underflow interrupt is disabled. 1 _B Overflow/underflow interrupt is enabled.
0	[7:2]	r	Reserved Returns 0 if read; should be written with 0.

14.8.3 Timer 2 Reload/Capture Register

The RC2 register is used for a 16-bit reload of the timer count upon overflow or a capture of current timer count depending on the mode selected.

T2_RC2L
Timer 2 Reload/Capture Register, Low Byte(C2_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


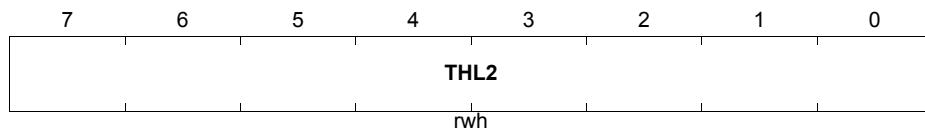
Field	Bit	Type	Description
RC2	[7:0]	rwh	Reload/Capture Value [7:0] If CP/RL2 = 0, these contents are loaded into the timer register upon an overflow condition. If CP/RL2 = 1, this register is loaded with the current timer count upon a negative/positive transition at pin T2EX when EXEN2 = 1.

T2_RC2H
Timer 2 Reload/Capture Register, Low Byte(C3_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


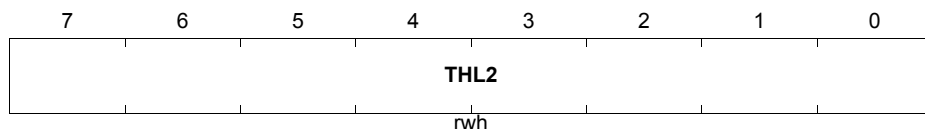
Field	Bit	Type	Description
RC2	[7:0]	rwh	Reload/Capture Value [15:8] If CP/RL2 = 0, these contents are loaded into the timer register upon an overflow condition. If CP/RL2 = 1, this register is loaded with the current timer count upon a negative/positive transition at pin T2EX when EXEN2 = 1.

14.8.4 Timer 2 Count Register

Register T2L and T2L hold the current 16-bit value of the Timer 2 count.

T2_T2L
Timer 2, Low Byte
(C4_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


Field	Bit	Type	Description
THL2	[7:0]	rwh	Timer 2 Value [7:0] These bits indicate the current 16-bit timer value.

T2_T2H
Timer 2, High Byte
(C5_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


Field	Bit	Type	Description
THL2	[7:0]	rwh	Timer 2 Value [15:8] These bits indicate the current 16-bit timer value.

15 Real-Time Clock

15.1 Overview

One of the XC82x's peripherals is the Real-Time Clock (RTC) that, once started, can work independently of the state of the rest of the microcontroller. There are two possible clock sources for this real-time clock, mainly the 75 KHz internal oscillator and an external clock input via RTCCLK pin.

Features

- Periodic Wake-up Mode with 75 KHz internal oscillator
- Wake-up source during power down mode

15.2 System Information

This section provides system information relevant to the Real-Time Clock.

15.2.1 Pinning

RTC clock source can be either from on-chip or an external source via a pin, RTCCLK. The RTCCLK pin assignment for XC82x is shown in [Table 15-1](#).

Table 15-1 RTC Pin Functions in XC82x

Pin	Function	Description	Selected By
P0.5	RTCCLK	RTC External Clock Input	–

15.2.2 Interrupt Events and Assignment

[Table 15-2](#) lists the interrupt event sources from the RTC, and the corresponding event interrupt enable bit and flag bit.

Table 15-2 RTC Interrupt Events

Event	Event Interrupt Enable Bit	Event Flag Bit
RTC compare/wake-up	RTC_RTCON.ECRTC	RTC_RTCON.CFRTC
RTC second time	RTC_RTCON.ESRTC	RTC_RTCON.SFRTC

[Table 15-3](#) shows the interrupt node assignment for each RTC interrupt source.

Table 15-3 RTC Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
RTC compare/wake-up	IEN1.EXM	–	4B _H
RTC second time			

15.2.3 Module Suspend Control

When the On-Chip Debug Support (OCDS) is in Monitor Mode (MMCR2.MMODE = 1) and the Debug-Suspend signal is active (MMCR2.DSUSP = 1), the timer/counter in RTC module in XC82x can be suspended based on the settings of their corresponding module suspend bits in register MODSUSP. The definition of this register is described in [Chapter 10.2.4](#).

When suspended, only the timer stops counting as the counter input clock is gated off. The module is still clocked so that module registers are accessible.

15.3 Oscillators

The 75 KHz internal oscillator can be used to clock the RTC. It is also used to clock an oscillation watchdog to monitor the 48 MHz oscillation which is the source of the system frequency.

Once power-up, the oscillator can operate irrespective of the state of the microcontroller. That is, it keeps running even when the device has entered idle or power down mode 2.

The oscillator, as well as the whole real-time clock, remains in operation during certain power down modes with a power supply of **2.5V - 5.5 V**.

15.4 Basic Timer Operation

The real-time clock consists of a 41-bit timer which count up. It contains a set of 4 to 6 count registers, collectively know at CNT register, that shows the current count value or the current time of the real-time clock . Before starting the real-time clock, CNT register can be written with any value. The value written is used as an initial value for the real-time clock when it is started. Another set of registers, RTCCR register, that consists of 4 to 6 registers can be used for interrupt generation. It can also be used to wake-up device from power down mode. The RTCCR register is also used to store the capture value when a capture event is triggered. The real-time clock is started by setting bit RTCC in the RTCON register to 1. This enables the input clock into the real-time clock timer.

15.5 Real-Time Clock Modes

In XC82x, the real-time clock operates in two modes. Mode 1 is the periodic wake-up mode that uses a 41-bit counter. In this mode, an internal oscillator of 75 KHz is used as

Real-Time Clock

the clock input. Mode 3 which has similar functions as Mode 1 uses an external clock input to a 32-bit counter. In XC82x, Mode 1 is selected by default upon power up. User need to ensure that the switching of mode is performed when the Real-time clock is in stop mode.

Note: In XC82x, there are only Mode 1 and 3. Mode 0 and Mode 2 are not available.

15.5.1 Mode 1: Periodic Wake-up Mode with 75 KHz Oscillator Clock

Figure 15-1 shows Mode 1 of the real-time clock. It is the default mode upon power on reset. In this mode, the real-time clock consists of a 41-bit general purposes timer. The 75 kHz oscillator is the input clock to the timer.

Before the real-time clock starts to run, CNT register of the real-time clock can be written with any values. The value written is used as an initial value for the real-time clock timer, when it is started by setting bit RTCC in the RTCON register to 1. This bit also enables the input clock into the real-time clock timer. CNT register is protected by the bit protection scheme as described in the SCU chapter. The initial count value can only be updated when the protection scheme is being disabled in the stop mode (RTCC = 0). The real-time clock's lower 9 bits, which serve as a prescaler into the 32-bit counters, CNT, are set to an initial value of 000000000_B when the RTC starts to run.

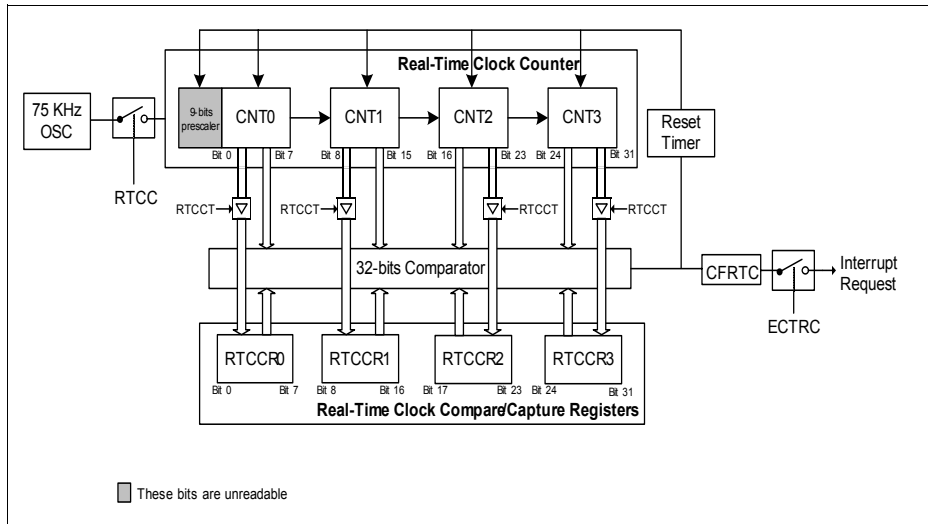


Figure 15-1 Real-time Clock Mode 1

While the real-time clock is in operation, the contents of RTCCR register will be compared to the real-time clock counter (CNT). An interrupt will be generated in active mode when the contents are equal if ECRTC is set to 1; bit CFRTC in register RTCON

Real-Time Clock

will be set. This will generate a wake-up from the software power-down when the device is in power down mode 2. The CFRTC flag can be monitored for the real-time clock wake-up request, but the flag has to be cleared by user software. In such situation, the real-time clock is reset and count from zero again. The handling of the wake-up is similar to the wake-up from power down mode through EXINT0 pin.

Note: No wake-up interrupt is generated after wake-up from power down mode without reset. It is regardless of the status of ECRTC bit. However, the CFRTC flag is set to 1 after wake-up in such situation.

With RTC mode 1 in power down mode 2, it is able to wake-up at a fixed time by programming the number of count value that is equivalent to the length of time to wake-up in the RTCCR register. The real-time clock can generate a wake-up request to the XC82x during power down mode provided all the following conditions are fulfilled:

- XC82x is in power down mode 2 (bit PDMODE = 1 in PMCON0 register),
- The power down mode is enabled (bit PD = 1 in PMCON0 register), and
- The type of wake-up mode is selected (bit WKSEL in PMCON0 register),
- Operating power supply levels (including reduced voltage conditions) are maintained

A capture event could be triggered by setting RTCCT bit in RTCON register to 1. The previous content in the RTCCR register will be overwritten with the captured CNT values after 2 CPU clock cycles. An update of the actual compared value is necessary once a captured event is triggered. In XC82x, it is recommended to trigger a capture event to read the value of the RTC counter (CNT). There is a potential of reading a wrong 32 bits real-time value while the RTC is in running mode as only 8 bit of data could be fetch at one time.

The real-time clock stops counting and CNT register holds the last value when the RTCC bit is set to 0. Setting this bit subsequently will start a new counting sequence which begin with the stop count.

Note: A compare match event could happen concurrently with the capture event when both events happen within the same clock cycle.

15.5.2 Mode 3: Timer Mode with External Clock

Figure 15-2 shows Mode 3 of the real-time clock. In this mode, the real-time clock consists of a 32-bit general purposes timer. It has the same function of Mode 1 except that the 9 bits prescaler is bypassed. The external clock via RTCCLK pin is the input clock to the timer. User need to select the RTCCLK pin to input mode and connect to a external clock before the counter can start to run.

In XC82x, RTC Mode 3 cannot be used to wake-up from power down mode.

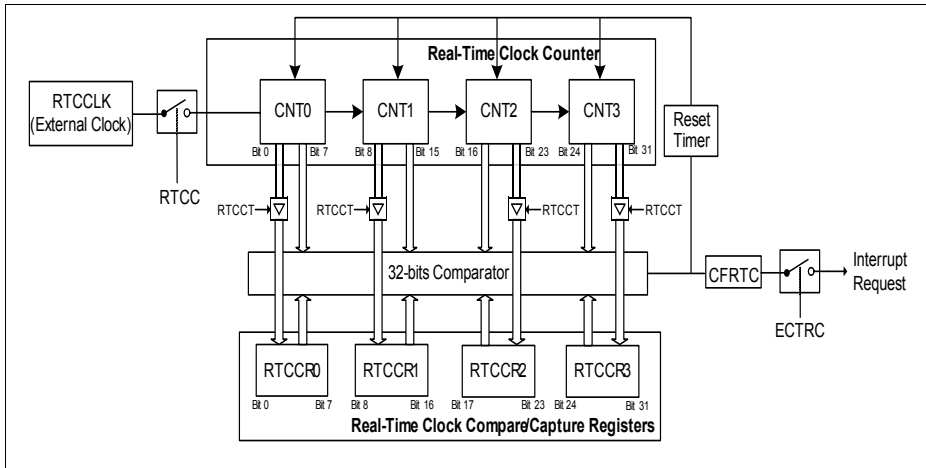


Figure 15-2 Real-time Clock Mode 3

15.6 Power Saving Mode Option

Once started, the real-time clock continues counting until the bit `RTCON.RTCC` is cleared. The real-time clock is not affected by the idle mode of the XC82x, and continues counting in power down mode except in power down mode 1. In addition, the real-time clock will not stopped automatically if the bit `OSC_CON.75KOSC2L` status is set to 1. In power-down modes 2, the real-time clock continues to run provided:

- the clock source for the real-time clock is available,
- $V_{DDP} > 2.5\text{ V}$,

The real-time clock stops operation in software power down mode 1.

15.7 Registers Description

14 SFRs are used to control the operation of real-time clock. They can be accessed from the standard (non-mapped) SFR area. The registers are described as follows.

Table 15-4 lists the addresses of these SFRs.

Table 15-4 Register Map

Address	Register
95 _H	RTCON
E1 _H	CNT0
E2 _H	CNT1
E3 _H	CNT2
E4 _H	CNT3
E7 _H	RTCCR0
E9 _H	RTCCR1
EA _H	RTCCR2
EB _H	RTCCR3

Real-Time Clock

15.7.1 Real-Time Clock Registers

RTC_RTCON

Real-Time Clock Control Register (95_H)

Reset Value: 02_H

RMAP: 0, PAGE: X

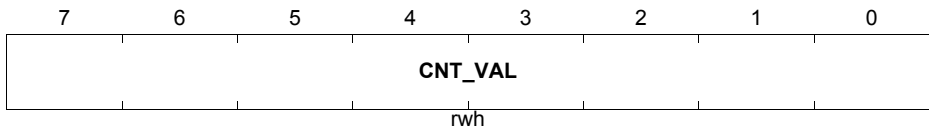
7	6	5	4	3	2	1	0
SFRTC	CFRTC	ESRTC	ECRTC	RTCCT	RTM	RTCC	
rwh	rwh	rw	rw	rwh	rw	rw	

Field	Bits	Type	Description
RTCC	0	rw	Real-Time Clock Start/Stop Control 0 _B Stop real-time clock Operation 1 _B Start real-time clock Operation
RTM	[2:1]	rw	Real-Time Clock Mode 0X _B Mode 1; Periodic wake-up mode with 75 KHz oscillator is selected. 1X _B Mode 3; Timer mode with direct external clock via RTCCLK is selected. <i>Note: Mode switching will caused the clock source to be switched at the same time.</i>
RTCCT	3	rwh	Real-Time Clock Capture Event Trigger 0 _B No action 1 _B Capture event is triggered immediately On set, this bit is held for two PCLK clock cycle, then cleared by hardware. Reading this bit always return 0.
ECRTC	4	rw	Real-Time Clock Compare Interrupt Enable 0 _B Disable real-time clock compare interrupt. 1 _B Enable real-time clock compare interrupt.
ESRTC	5	rw	Real-Time Clock Second Timing Interrupt Enable 0 _B Disable real-time clock interrupt at every second in Mode 0. 1 _B Enable real-time clock interrupt at every second in Mode 0. <i>Note: The interrupt function at every second is only available in Mode 0 and Mode 2.</i>

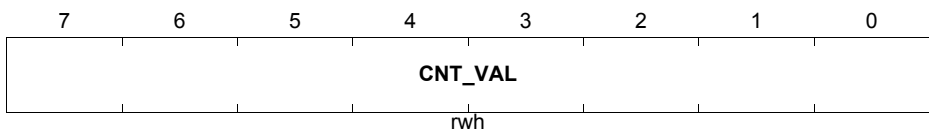
Real-Time Clock

Field	Bits	Type	Description
CFRTC	6	rwh	Real-Time Clock Compare Flag This bit is set by hardware when there is a compare match and can only be cleared by software. A wake-up request is generated only if the XC82x is either in power down mode 2, 3 or 4.
SFRTC	7	rwh	Real-Time Clock Second Timing Flag This bit is set by hardware at every second and can only be cleared by software. It is only available in Mode 0 and Mode 2.

4 count registers, CNT0 - CNT3 are used to represent the most significant 32 bits of the 41-bits real-time clock timer in Mode 1 and 32-bits timer in Mode 3.

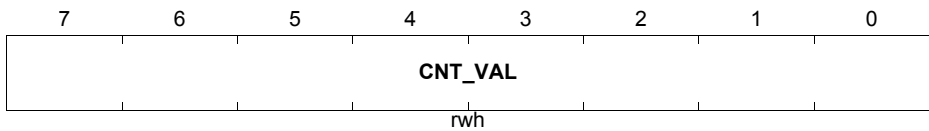
RTC_CNT0 [Mode 1 and Mode 3]
Count Clock Register 0
(E1_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


Field	Bits	Type	Description
CNT_VAL	[7:0]	rwh	Real-Time Clock Count Value [7:0] These bits represent counter value [7:0] of the current real-time clock timer.

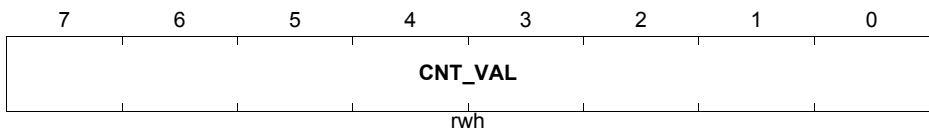
RTC_CNT1 [Mode 1 and Mode 3]
Count Clock Register 1
(E2_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


Real-Time Clock

Field	Bits	Type	Description
CNT_VAL	[7:0]	rwh	Real-Time Clock Count Value [15:8] These bits represent counter value [15:8] of the current real-time clock timer.

RTC_CNT2 [Mode 1 and Mode 3]
Count Clock Register 2
(E3_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


Field	Bits	Type	Description
CNT_VAL	[7:0]	rwh	Real-Time Clock Count Value [23:16] These bits represent counter value [23:16] of the current real-time clock timer.

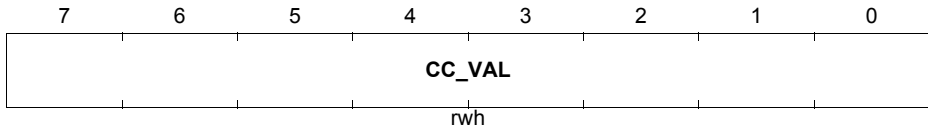
RTC_CNT3 [Mode 1 and Mode 3]
Count Clock Register 3
(E4_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


Field	Bits	Type	Description
CNT_VAL	[7:0]	rwh	Real-Time Clock Count Value [31:23] These bits represent counter value [31:23] of the current real-time clock timer.

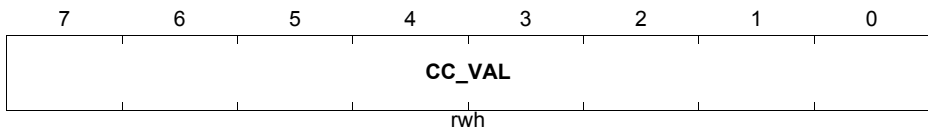
In Mode 1 and Mode 3, 4 compare and capture registers, RTCCR0 - RTCCR3 are used to represent 32 bits of compare values that will generate a compare event when it matches the counter values as specified in CNT register. When a capture event is triggered by setting RTCCT bit to 1, the content in the RTCCR register will be overwritten

Real-Time Clock

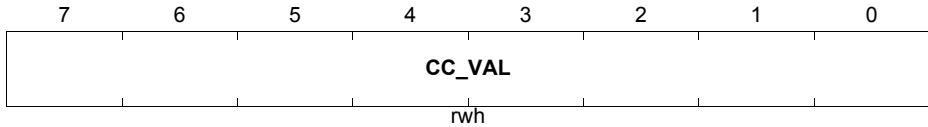
with the captured CNT values after 2 CPU clock cycles. An update of the actual compared value is necessary once a captured event is triggered.

RTC_RTCCR0 [Mode 1 and Mode 3]
Real-Time Clock Compare/Capture Register 0(E7_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


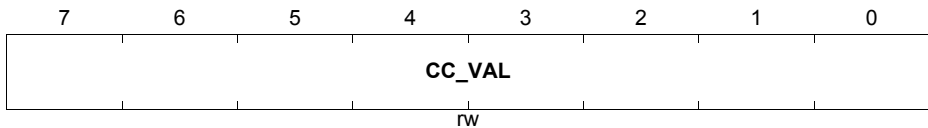
Field	Bits	Type	Description
CC_VAL	[7:0]	rwh	Compare/Capture Value [7:0] These bits represent the compare/capture value [7:0] of the 32-bits value that could generate a compare interrupt when it matches with the current counter values.

RTC_RTCCR1 [Mode 1 and Mode 3]
Real-Time Clock Compare/Capture Register 1(E9_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


Field	Bits	Type	Description
CC_VAL	[7:0]	rwh	Compare/Capture Value [15:8] These bits represent compare/capture value [15:8] of the 32-bits value that could generate a compare interrupt when it matches with the current counter values.

Real-Time Clock
RTC_RTCCR2 [Mode 1 and Mode 3]
Real-Time Clock Compare/Capture Register 2(EA_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


Field	Bits	Type	Description
CC_VAL	[7:0]	rwh	Compare/Capture Value [23:16] These bits represent compare/capture value [23:16] of the 32-bits value that could generate a compare interrupt when it matches with the current counter value.

RTC_RTCCR3 [Mode 1 and Mode 3]
Real-Time Clock Compare/Capture Register 3(EB_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


Field	Bits	Type	Description
CC_VAL	[7:0]	rwh	Compare/Capture Value [31:23] These bits represent compare/capture value [31:23] of the 32-bits value that could generate a compare interrupt when it matches with the current counter value.

16 UART

16.1 Overview

The UART provides a full-duplex asynchronous receiver/transmitter, i.e., it can transmit and receive simultaneously. It is also receive-buffered, i.e., it can commence reception of a second byte before a previously received byte has been read from the receive register. However, if the first byte still has not been read by the time reception of the second byte is complete, one of the bytes will be lost.

UART Feature:

- Full-duplex asynchronous modes
 - 8-bit or 9-bit data frames, LSB first
 - fixed or variable baud rate
- Receive buffered
- Multiprocessor communication
- Interrupt generation on the completion of a data transmission or reception

16.2 System Information

This section provides system information relevant to the UART.

16.2.1 Pinning

In mode 0 (the serial port behaves as shift register), data is shifted in through RXD and out through RXDO, while the TXD line is used to provide a shift clock which can be used by external devices to clock data in and out. In modes 1, 2 and 3, the port behaves as an UART. Data is transmitted on TXD and received on RXD.

Note: XC82x does not support mode 0 operation and therefore, RXDO line is not connected to any general purpose I/O port.

The UART I/O pins are generally assigned as alternate functions to general purpose I/O ports. Bit URRIS in register MODPISEL1 is used to select one of the RXD input function. The UART pin assignment for XC82x is shown in [Table 16-1](#).

Table 16-1 UART Pin Functions in XC82x

Pin	Function	Description	Selected By
P0.5	RXD_0	UART Receive Data Input 0	MODPISEL1.URRIS = 00 _B
P0.6	RXD_1	UART Receive Data Input 1	MODPISEL1.URRIS = 01 _B
P1.0	RXD_2	UART Receive Data Input 2	MODPISEL1.URRIS = 10 _B

Table 16-1 UART Pin Functions in XC82x

Pin	Function	Description	Selected By
P2.1	RXD_3	UART Receive Data Input 3	MODPISEL1.URRIS = 11 _B
P0.6	TXD_0	UART Transmit Data Output 0	P0_ALTSEL0.P5 = 1 _B P0_ALTSEL1.P5 = 1 _B P0_ALTSEL2.P5 = 0 _B
P1.0	TXD_1	UART Transmit Data Output 1	P1_ALTSEL0.P0 = 1 _B P1_ALTSEL1.P0 = 1 _B
P1.1	TXD_2	UART Transmit Data Output 2	P1_ALTSEL0.P1 = 1 _B P1_ALTSEL1.P1 = 1 _B
P0.5	TXD_3	UART Transmit Data Output 4	P0_ALTSEL0.P5 = 0 _B P0_ALTSEL1.P5 = 0 _B P0_ALTSEL2.P5 = 1 _B

The bit field PAGE of SCU_PAGE register must be programmed before accessing the MODPISEL1 register.

MODPISEL1
Peripheral Input Select Register 1 (F4_H)
Reset Value: 00_H
RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	EXINT1IS	0	EXINT0IS		0		URRIS
r	rw	r	rw		r		rw

Field	Bits	Type	Description
URRIS	[1:0]	rw	UART Receive Input Select 00 UART Receiver Input RXD_0 is selected. 01 UART Receiver Input RXD_1 is selected. 10 UART Receiver Input RXD_2 is selected. 11 UART Receiver Input RXD_3 is selected.
0	2, 5, 7	r	Reserved Returns 0 if read; should be written with 0.

16.2.2 Clocking Configuration

The UART runs on the PCLK at a frequency of either 8 MHz or 24 MHz.

16.2.3 Interrupt Events and Assignment

Table 16-2 lists the interrupt event sources from the UART, and the corresponding event interrupt enable bit and flag bit.

Table 16-2 UART Interrupt Events

Event	Event Interrupt Enable Bit	Event Flag Bit
Data Received	–	SCON.RI
Data Transmitted		SCON.TI

Table 16-3 shows the interrupt node assignment for each UART interrupt source.

Table 16-3 UART Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
Data Received	IEN0.ES	–	23 _H
Data Transmitted			

16.3 UART Modes

The UART can be used in four different modes. In mode 0, it operates as an 8-bit shift register. In mode 1, it operates as an 8-bit serial port. In modes 2 and 3, it operates as a 9-bit serial port. The only difference between mode 2 and mode 3 is the baud rate, which is fixed in mode 2 but variable in mode 3. The variable baud rate is set by either the underflow rate on the dedicated baud-rate generator, or by the overflow rate on Timer 1. The different modes are selected by setting bits SM0 and SM1 to their corresponding values, as shown in **Table 16-4**.

Table 16-4 UART Modes

SM0	SM1	Operating Mode	Baud Rate
0	0	Mode 0: 8-bit shift register	$f_{PCLK}/2$
0	1	Mode 1: 8-bit shift UART	Variable
1	0	Mode 2: 9-bit shift UART	$f_{PCLK}/64$ or $f_{PCLK}/32$
1	1	Mode 3: 9-bit shift UART	Variable

16.3.1 Mode 0, 8-Bit Shift Register, Fixed Baud Rate

In mode 0, the serial port behaves as an 8-bit shift register. Data is shifted in through RXD, and out through RXDO, while the TXD line is used to provide a shift clock which can be used by external devices to clock data in and out.

UART

The transmission cycle is activated by a write to SBUF. One machine cycle later, the data has been written to the transmit shift register with a 1 at the 9th bit position. For the next seven machine cycles, the contents of the transmit shift register are shifted right one position and a zero shifted in from the left so that when the MSB of the data byte is at the output position, it has a 1 and a sequence of zeros to its left. The control block then executes one last shift before resetting the TI bit.

Reception is started by the condition $REN = 1$ and $RI = 0$. At the start of the reception cycle, 11111110_B is written to the receive shift register. In each machine cycle that follows, the contents of the shift register are shifted left one position and the value sampled on the RXD line in the same machine cycle is shifted in from the right. When the 0 of the initial byte reaches the leftmost position, the control block executes one last shift, loads SBUF and sets the RI bit.

The baud rate for the transfer is fixed at $f_{PCLK}/2$ where f_{PCLK} is the input clock frequency, i.e. one bit per machine cycle.

16.3.2 Mode 1, 8-Bit UART, Variable Baud Rate

In mode 1, the UART behaves as an 8-bit serial port. A start bit (0), 8 data bits, and a stop bit (1) are transmitted on TXD or received on RXD at a variable baud rate.

The transmission cycle is activated by a write to SBUF. The data is transferred to the transmit register and a 1 is loaded to the 9th bit position (as in mode 0). At phase 1 of the machine cycle after the next rollover in the divide-by-16 counter, the start bit is copied to TXD, and data is activated one bit time later. One bit time after the data is activated, the data starts getting shifted right with zeros shifted in from the left. When the MSB gets to the output position, the control block executes one last shift and sets the TI bit.

Reception is started by a high to low transition on RXD (sampled at 16 times the baud rate). The divide-by-16 counter is then reset and $1111\ 1111_B$ is written to the receive register. If a valid start bit (0) is then detected (based on two out of three samples), it is shifted into the register followed by 8 data bits. If the transition is not followed by a valid start bit, the controller goes back to looking for a high to low transition on RXD. When the start bit reaches the leftmost position, the control block executes one last shift, then loads SBUF with the 8 data bits, loads RB8 (SCON.2) with the stop bit, and sets the RI bit, provided $RI = 0$, and either $SM2 = 0$ (see [Section 16.4](#)) or the received stop bit = 1. If none of these conditions is met, the received byte is lost.

The associated timings for transmit/receive in mode 1 are illustrated in [Figure 16-1](#).

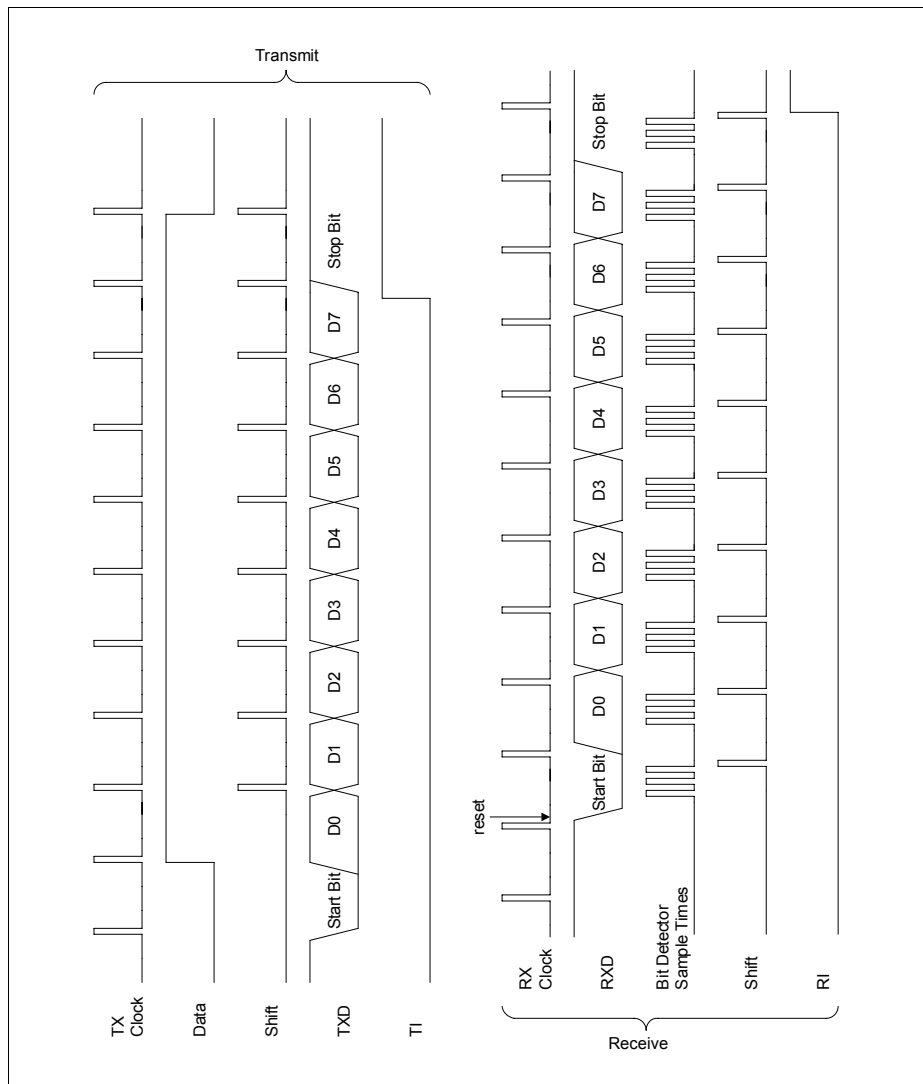


Figure 16-1 Serial Interface, Mode 1, Timing Diagram

16.3.3 Mode 2, 9-Bit UART, Fixed Baud Rate

In mode 2, the UART behaves as a 9-bit serial port. A start bit (0), 8 data bits plus a programmable 9th bit and a stop bit (1) are transmitted on TXD or received on RXD. The 9th bit for transmission is taken from TB8 (SCON.3) while for reception, the 9th bit received is placed in RB8 (SCON.2).

The transmission cycle is activated by a write to SBUF. The data is transferred to the transmit register and TB8 is copied into the 9th bit position. At phase 1 of the machine cycle following the next rollover in the divide-by-16 counter, the start bit is copied to TXD and data is activated one bit time later. One bit time after the data is activated, the data starts shifting right. For the first shift, a stop bit (1) is shifted in from the left and for subsequent shifts, zeros are shifted in. When the TB8 bit gets to the output position, the control block executes one last shift and sets the TI bit.

Reception is started by a high to low transition on RXD (sampled at 16 times the baud rate). The divide-by-16 counter is then reset and 1111 1111_B is written to the receive register. If a valid start bit (0) is then detected (based on two out of three samples), it is shifted into the register followed by 8 data bits. If the transition is not followed by a valid start bit, the controller goes back to looking for a high to low transition on RXD. When the start bit reaches the leftmost position, the control block executes one last shift, then loads SBUF with the 8 data bits, loads RB8 (SCON.2) with the 9th data bit, and sets the RI bit, provided RI = 0, and either SM2 = 0 (see [Section 16.4](#)) or the 9th bit = 1. If none of these conditions is met, the received byte is lost.

The baud rate for the transfer is either $f_{PCLK}/64$ or $f_{PCLK}/32$, depending on the setting of the top bit (SMOD) of the PCON (Power Control) register, which acts as a Double Baud Rate selector.

16.3.4 Mode 3, 9-Bit UART, Variable Baud Rate

Mode 3 is the same as mode 2 in all respects except that the baud rate is variable.

In all modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in the modes by the incoming start bit if REN = 1.

The serial interface also provides interrupt requests when transmission or reception of the frames has been completed. The corresponding interrupt request flags are TI or RI, respectively. If the serial interrupt is not used (i.e., serial interrupt not enabled), TI and RI can also be used for polling the serial interface.

The associated timings for transmit/receive in modes 2 and 3 are illustrated in [Figure 16-2](#).

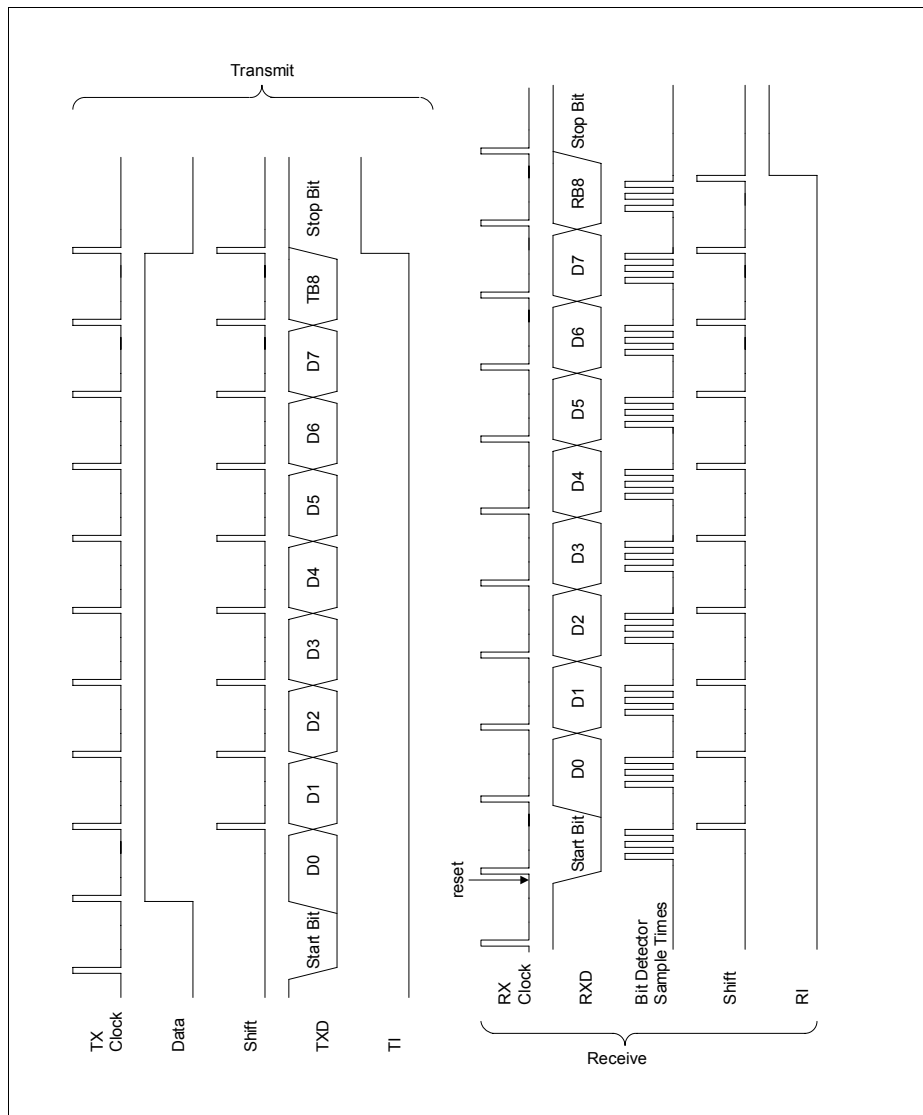


Figure 16-2 Serial Interface, Modes 2 and 3, Timing Diagram

16.4 Multiprocessor Communication

Modes 2 and 3 have a special provision for multiprocessor communication using a system of address bytes with bit 9 = 1 and data bytes with bit 9 = 0. In these modes, 9 data bits are received. The 9th data bit goes into RB8. The communication always ends with one stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8 = 1.

This feature is enabled by setting bit SM2 in SCON. One of the ways to use this feature in multiprocessor systems is described in the following paragraph.

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte that identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2 = 1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and see if it is being addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that were not being addressed retain their SM2s as set and ignore the incoming data bytes.

Bit SM2 has no effect in mode 0. SM2 can be used in mode 1 to check the validity of the stop bit. In a mode 1 reception, if SM2 = 1, the receive interrupt will not be activated unless a valid stop bit is received.

16.5 Baud Rate Generation

There are several ways to generate the baud rate clock for the serial ports, depending on the mode in which they are operating.

The baud rates in modes 0 and 2 are fixed, so they use the

- Fixed clock, (see [Section 16.5.1](#))

In modes 1 and 3, the variable baud rate is generated using either the

- UART baud-rate generator (see [Section 16.5.2](#)); or
- Timer 1 (see [Section 16.5.3](#))

This selection between the different variable baud rate sources is performed by bit BGS in LINST register.

“Baud rate clock” and “baud rate” must be distinguished from each other. The serial interface requires a clock rate that is 16 times the baud rate for internal synchronization. Therefore, the UART baud-rate generator and Timer 1 must provide a “baud rate clock” to the serial interface where it is divided by 16 to obtain the actual “baud rate”. The abbreviation f_{PCLK} refers to the input clock frequency.

16.5.1 Fixed Clock

The baud rates in modes 0 and 2 are fixed. However, while the baud rate in mode 0 can only be $f_{PCLK}/2$, the baud rate in mode 2 can be selected as either $f_{PCLK}/64$ or $f_{PCLK}/32$ depending on bit SMOD in the PCON register.

Bit SMOD acts as a double baud rate selector, as shown in [Equation \(16.1\)](#). If SMOD = 0 (value after reset), the baud rate is 1/64 of the input clock frequency f_{PCLK} . If SMOD = 1, the baud rate is 1/32 of f_{PCLK} .

(16.1)

$$\text{Mode 2 baud rate} = \frac{2^{\text{SMOD}}}{64} \times f_{PCLK}$$

16.5.2 UART Baud-rate Generator

The UART baud-rate generator is used to generate the variable baud rate for the UART in modes 1 and 3. It has programmable 11-bit reload value, 3-bit prescaler and 5-bit fractional divider.

The baud-rate generator is clocked derived via a prescaler (f_{DIV}) from the input clock f_{PCLK} . The baud rate timer counts downwards and can be started or stopped through the baud rate control run bit BCON.R. Each underflow of the timer provides one clock pulse to the serial channel. The timer is reloaded with the 11-bit BR_VALUE stored in its reload

UART

register BG each time it underflows. The duration between underflows depends on the 'n' value in the fractional divider, which can be selected by the bits BGL.FD_SEL. 'n' times out of 32, the timer counts one cycle more than specified by BR_VALUE. The prescaler is selected by the bits BCON.BRPRE.

Register BG is the dual-function Baud-rate Generator/Reload register. Reading BG returns the contents of the timer, while writing to BG (low byte) always updates the reload register.

The BG should be written only when BCON.R is 0. An auto-reload of the timer with the contents of the reload register is performed one instruction cycle after the next time BCON.R is set. Any write to BG, while BCON.R is set, will be ignored.

The baud rate of the baud-rate generator depends on the following bits and register values:

- Input clock f_{PCLK}
- Value of bit field BCON.BRPRE.
- Value of bit field BG.FD_SEL
- Value of the 11-bit reload value BG.BR_VALUE

Figure 16-3 shows a simplified block diagram of the baud rate generator.

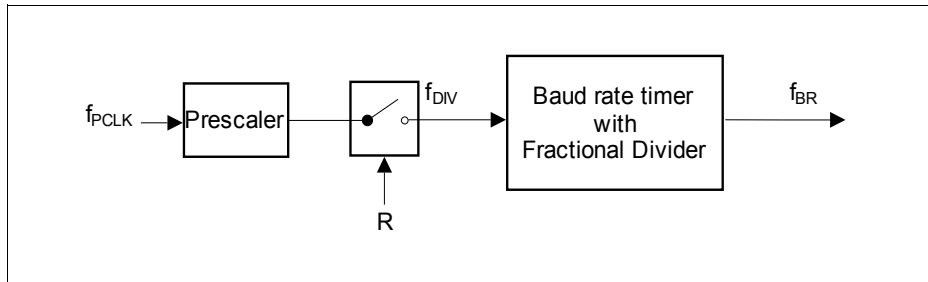


Figure 16-3 Simplified Baud Rate Generator Block Diagram

The following formula calculate the final baud rate.

(16.2)

$$\text{Baud rate} = \frac{f_{PCLK}}{16 \times \text{PRE} \times \left(\text{BR_VALUE} + \frac{n}{32} \right)}$$

The value of PRE (prescaler) is chosen by the bit field BCON.BRPRE. BR_VALUE represents the contents of the reload value, taken as unsigned 11-bit integer from the bit

UART

field BG.BR_VALUE. $n/32$ is defined by the fractional divider selection in bit field BG.FDSEL.

The maximum baud rate that can be generated is limited to $f_{PCLK}/32$. Hence, for module clocks of 8 MHz and 24 MHz, the maximum achievable baud rate is 0.25 MBaud and 0.75 MBaud respectively.

Table 16-5 and **Table 16-6** list various commonly used baud rates together with their corresponding parameter settings and the deviation errors compared to the intended baud rate.

Table 16-5 Typical Baud Rates of UART ($f_{PCLK} = 8 \text{ MHz}$)

Baud rate ($f_{PCLK} = 8 \text{ MHz}$)	PRE	Reload Value (BR_VALUE)	Numerator of Fractional Value (FD_NUM)	BG Register ¹⁾	Deviation Error
115.2 kBaud	1 (BRPRE = 000)	4 (4 _H)	11 (B _H)	008B _H	-0.08%
20 kBaud	1 (BRPRE = 000)	25 (19 _H)	0 (0 _H)	0320 _H	0.00%
19.2 kBaud	1 (BRPRE = 000)	26 (1A _H)	1 (1 _H)	0341 _H	+0.04%
9600 Baud	2 (BRPRE = 001)	26 (1A _H)	1 (1 _H)	0341 _H	+0.04%
4800 Baud	4 (BRPRE = 010)	26 (1A _H)	1 (1 _H)	0341 _H	+0.04%
2400 Baud	8 (BRPRE = 011)	26 (1A _H)	1 (1 _H)	0341 _H	+0.04%

1) The value of the 16-bit BG register is obtained by concatenating the 11-bit BRVALUE and 5-bit FD_NUM into a 16-bit value.

Table 16-6 Typical Baud Rates of UART ($f_{PCLK} = 24 \text{ MHz}$)

Baud rate ($f_{PCLK} = 24 \text{ MHz}$)	PRE	Reload Value (BR_VALUE)	Numerator of Fractional Value (FD_NUM)	BG Register ¹⁾	Deviation Error
115.2 kBaud	1 (BRPRE = 000)	13 (0D _H)	1 (01 _H)	01A1 _H	-0.08%
20 kBaud	1 (BRPRE = 000)	75 (4B _H)	0 (00 _H)	0960 _H	+0.00%
19.2 kBaud	1 (BRPRE = 000)	78 (4E _H)	4 (04 _H)	09C4 _H	+0.00%
9600 Baud	2 (BRPRE = 001)	78 (4E _H)	4 (04 _H)	09C4 _H	+0.00%
4800 Baud	4 (BRPRE = 010)	78 (4E _H)	4 (04 _H)	09C4 _H	+0.00%
2400 Baud	8 (BRPRE = 011)	78 (4E _H)	4 (04 _H)	09C4 _H	+0.00%

1) The value of the 16-bit BG register is obtained by concatenating the 11-bit BRVALUE and 5-bit FD_NUM into a 16-bit value.

16.5.3 Timer 1

In modes 1 and 3 of UART, Timer 1 can also be used for generating the variable baud rates. In theory, this timer could be used in any of its modes. But in practice, it should be set into auto-reload mode (Timer 1 mode 2), with its high byte set to the appropriate value for the required baud rate. The baud rate is determined by the Timer 1 overflow rate and the value of SMOD bit in the PCON register as follows:

(16.3)

$$\text{Mode 1, 3 baud rate} = \frac{2^{\text{SMOD}} \times f_{\text{PCLK}}}{32 \times 2 \times (256 - \text{TH1})}$$

Alternatively, for a given baud rate, the value of Timer 1 high byte can be derived:

(16.4)

$$\text{TH1} = 256 - \frac{2^{\text{SMOD}} \times f_{\text{PCLK}}}{32 \times 2 \times \text{Mode 1, 3 baud rate}}$$

Note: Timer 1 can neither indicate an overflow nor generate an interrupt if Timer 0 is in mode 3; Timer 1 is halted while Timer 0 takes over the use of its control bits and overflow flag. Hence, the baud rate supplied to the UART is defined by Timer 0 and not Timer 1. User should avoid using Timer 0 and Timer 1 in mode 3 for baud rate generation.

16.6 LIN Support in UART

The UART module can be used to support the Local Interconnect Network (LIN) protocol for both master and slave operations. The LIN baud rate detection feature, which consists of the hardware logic for Break and Synch Field detection, provides the capability to detect the baud rate within LIN protocol using Timer 2. This allows the UART module to be synchronized to the LIN baud rate for data transmission and reception.

16.6.1 LIN Protocol

LIN is a holistic communication concept for local interconnected networks in vehicles. The communication is based on the SCI (UART) data format, a single-master/multiple-slave concept, a clock synchronization for nodes without stabilized time base. An attractive feature of LIN is self-synchronization of the slave nodes without a crystal or ceramic resonator, which significantly reduces the cost of hardware platform. Hence, the baud rate must be calculated and returned with every message frame.

The structure of a LIN frame is shown in [Figure 16-4](#). The frame consists of the:

- header, which comprises a Break (13-bit time low), Synch Byte (55_H), and ID field
- response time
- data bytes (according to UART protocol)
- checksum

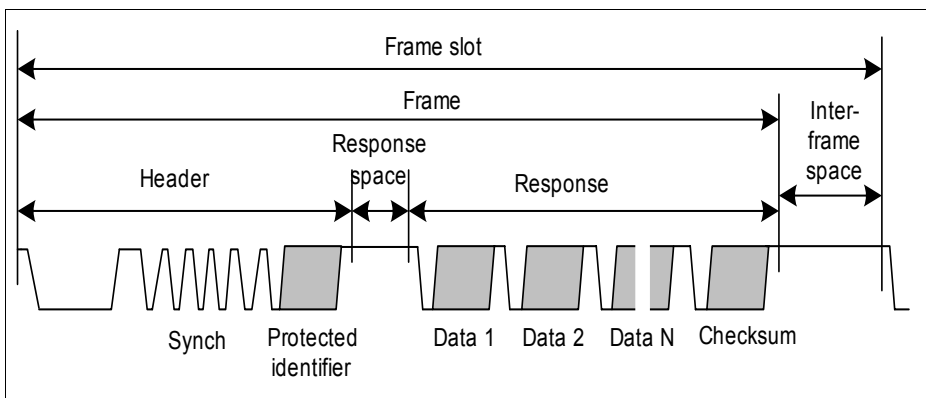


Figure 16-4 The Structure of LIN Frame

Each byte field is transmitted as a serial byte, as shown in [Figure 16-5](#). The LSB of the data is sent first and the MSB is sent last. The start bit is encoded as a bit with value zero (dominant) and the stop bit is encoded as a bit with value one (recessive).

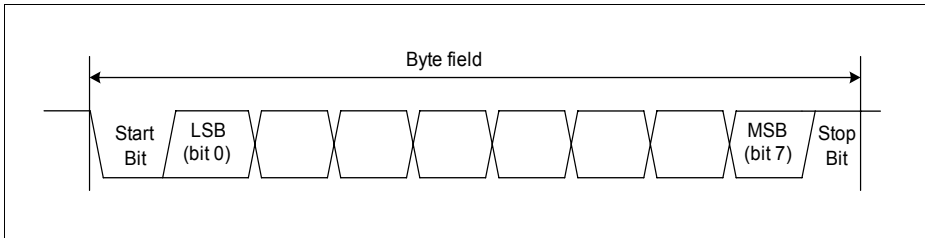


Figure 16-5 The Structure of Byte Field

The break is used to signal the beginning of a new frame. It is the only field that does not comply with [Figure 16-5](#). A break is always generated by the master task (in the master mode) and it must be at least 13 bits of dominant value, including the start bit, followed by a break delimiter, as shown in [Figure 16-6](#). The break delimiter will be at least one nominal bit time long.

A slave node will use a break detection threshold of 11 nominal bit times.

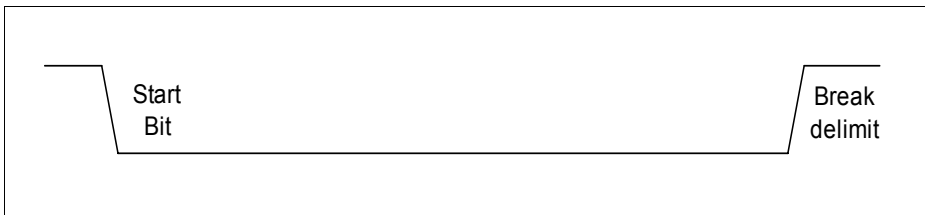


Figure 16-6 The Break Field

Synch Byte is a specific pattern for determination of time base. The byte field is with the data value 55_H, as shown in [Figure 16-7](#).

A slave task is always able to detect the Break/Synch sequence, even if it expects a byte field (assuming the byte fields are separated from each other). If this happens, detection of the Break/Synch sequence will abort the transfer in progress and processing of the new frame will commence.

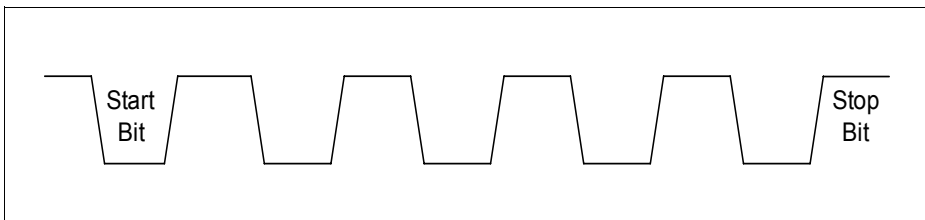


Figure 16-7 The Synch Byte Field

The slave task will receive and transmit data when an appropriate ID is sent by the master:

1. Slave waits for Synch Break
2. Slave synchronizes on Synch Byte
3. Slave snoops for ID
4. According to ID, slave determines whether to receive or transmit data, or do nothing
5. When transmitting, the slave sends 2, 4 or 8 data bytes, followed by check byte

16.6.2 LIN Header Transmission

LIN header transmission is only applicable in master mode. In the LIN communication, a master task decides when and which frame is to be transferred on the bus. It also identifies a slave task to provide the data transported by each frame. The information needed for the handshaking between the master and slave tasks is provided by the master task through the header portion of the frame.

The header consists of a break and synch pattern followed by an identifier. Among these three fields, only the break pattern cannot be transmitted as a normal 8-bit UART data. The break must contain a dominant value of 13 bits or more to ensure proper synchronization of slave nodes.

In the LIN communication, a slave task is required to be synchronized at the beginning of the protected identifier field of frame. For this purpose, every frame starts with a sequence consisting of a break field followed by a synch byte field. This sequence is unique and provides enough information for any slave task to detect the beginning of a new frame and be synchronized at the start of the identifier field.

16.6.2.1 Automatic Synchronization to the Host

Upon entering LIN communication, a connection is established and the transfer speed (baud rate) of the serial communication partner (host) is automatically synchronized in the following steps that are to be included in user software:

STEP 1: Initialize interface for reception and timer for baud rate measurement

STEP 2: Wait for an incoming LIN frame from host

STEP 3: Synchronize the baud rate to the host

STEP 4: Enter for Master Request Frame or for Slave Response Frame

The next section, [Section 16.6.2.2](#), provides some hints on setting up the microcontroller for baud rate detection of LIN.

*Note: Re-synchronization and setup of baud rate are always done for **every** Master Request Header or Slave Response Header LIN frame.*

16.6.2.2 Initialization of Break/Synch Field Detection Logic

The LIN baud rate detection feature provides the capability to detect the baud rate within the LIN protocol using Timer 2. Initialization consists of:

- Serial port of the microcontroller set to Mode 1 (8-bit UART, variable baud rate) for communication.
- Provide the baud rate range via bit field BCON.BGSEL.
- Toggle BCON.BRDIS bit (set the bit to 1 before clearing it back to 0) to initialize the Break/Synch detection logic.
- Clear all status flags LINST.BRK, LINST.EOFSYN and LINST.ERRSYN to 0.
- Timer 2 is set to capture mode with falling edge trigger at pin T2EX. Bit T2MOD.EDGESEL is set to 0 by default and bit T2CON.CP/RL2 is set to 1.
- Timer 2 external events are enabled. T2CON.EXEN2 is set to 1. (EXF2 flag is set when a negative transition occurs at pin T2EX)
- f_{T2} can be configured by bit field T2MOD.T2PRE.

Note: It is also recommended to toggle the BCON.BRDIS bit after the reception of each complete LIN frame to avoid a wrong Break field detection in noisy environments (i.e. spikes on the LIN bus).

16.6.2.3 Baud Rate Range Selection

The Break/Synch Field detection logic supports a maximum number of bits in the Break field as defined by [Equation \(16.5\)](#).

(16.5)

$$\text{Maximum number of bits} = \text{Baud Rate} \times \frac{4095}{\text{Sample Frequency}}$$

The sample frequency is given by [Equation \(16.6\)](#).

(16.6)

$$\text{Sample Frequency} = \frac{\text{PCLK}}{8 \times 2^{\text{BGSEL}}}$$

If the maximum number of bits in the Break field is exceeded, the internal counter will overflow, which results in a baudrate detection error. Therefore, an appropriate BGSEL value has to be selected for the required baudrate detection range.

The baud rate range defined by different BGSEL settings is shown in [Table 16-7](#).

Table 16-7 BGSEL Bit Field Definition for Different Input Frequencies

f_{PCLK}	BGSEL	Baud Rate Select for Detection $f_{\text{pclk}}/(2184 \cdot 2^{\text{BGSEL}})$ to $f_{\text{pclk}}/(72 \cdot 2^{\text{BGSEL}})$
24 MHz	00 _B	11 kHz to 333.3 kHz
	01 _B	5.5 kHz to 166.7 kHz
	10 _B	2.8 kHz to 83.3 kHz
	11 _B	1.4 kHz to 41.7 kHz
8 MHz	00 _B	3.7 kHz to 111.1 kHz
	01 _B	1.8 kHz to 55.6 kHz
	10 _B	0.92 kHz to 27.8 kHz
	11 _B	0.46 kHz to 13.9 kHz

Each BGSEL setting supports a range of baud rate for detection. If the baud rate used is outside the defined range, the baud rate may not be detected correctly.

When $f_{\text{pclk}} = 24$ MHz, the baud rate range between 1.4 kHz to 333.3 kHz can be detected. The following examples serve as a guide to select BGSEL value:

- If the baud rate falls in the range of 1.4 kHz to 2.8 kHz, selected BGSEL value is "11_B".
- If the baud rate falls in the range of 2.8 kHz to 5.5 kHz, selected BGSEL value is "10_B".
- If the baud rate falls in the range of 5.5 kHz to 11 kHz, selected BGSEL value is "01_B".
- If the baud rate falls in the range of 11 kHz to 333.3 kHz, selected BGSEL value is "00_B". If the baud rate is 20 kHz, the possible values of BGSEL that can be selected are "00_B", "01_B", "10_B", and "11_B". However, it is advisable to select "00_B" for better detection accuracy.

The baud rate can also be detected when $f_{\text{pclk}} = 8$ MHz, for which the baud rate range that can be detected is between 0.46 kHz to 111.1 kHz.

16.6.2.4 LIN Baud Rate Detection

The baud rate detection for LIN is shown in **Figure 16-8**, the Header LIN frame consists of the:

- SYN Break (13 bit times low)
- SYN byte (55_H)
- Protected ID field

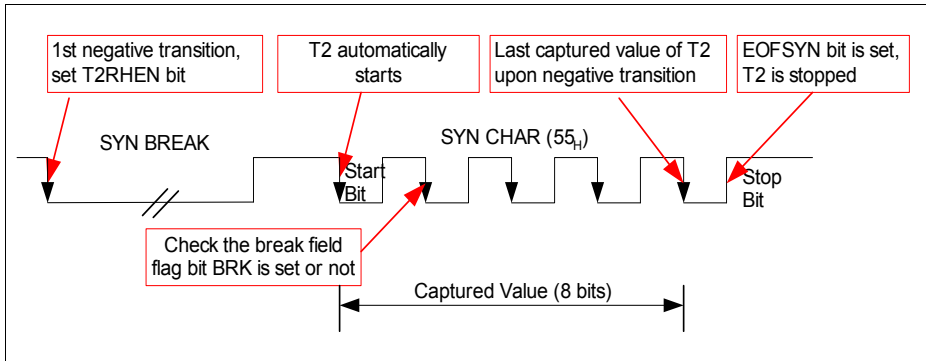


Figure 16-8 LIN Auto Baud Rate Detection

With the first falling edge:

- The Timer 2 External Start Enable bit (T2MOD.T2RHEN) is set. The falling edge at pin T2EX is selected by default for Timer 2 External Start (bit T2MOD.T2REGS is 0).

With the second falling edge:

- Start Timer 2 by the hardware.

With the third falling edge:

- Timer 2 captures the timing of 2 bits of SYN byte.
- Check the Break Field Flag bit LINST.BRK.

If the Break Field Flag LINST.BRK is set, software may continue to capture 4/6/8 bits of SYN byte. Finally, the End of SYN Byte Flag (LINST.EOFSYN) is set, Timer 2 is stopped. T2 Reload/Capture register (RC2H/L) is the time taken for 2/4/6/8 bits according to the implementation. Then the LIN routine calculates the actual baud rate, sets the PRE and BG values if the UART module uses the baud-rate generator for baud rate generation.

After the third falling edge, the software may discard the current operation and continue to detect the next header LIN frame if the following conditions were detected:

- The Break Field Flag LINST.BRK is not set, or
- The SYN Byte Error Flag LINST.ERRSYN is set, or
- The Break Field Flag LINST.BRK is set, but the End of SYN Byte Flag LINST.EOFSYN and the SYN Byte Error Flag LINST.ERRSYN are not set.

16.7 Registers Description

Besides the SCON and SBUF registers, which can be accessed from both the standard (non-mapped) and mapped SFR area, the rest of the UART's SFRs are located in SCU page 5 of the standard area. The bit field PAGE of SCU_PAGE register must be programmed before accessing these registers.

Table 16-8 lists the addresses of these SFRs.

Table 16-8 Register Map

Address	Register
98 _H	SCON
99 _H	SBUF
F2 _H	BCON
F3 _H	BGL
F4 _H	BGH
F5 _H	LINST

16.7.1 UART Registers

UART contains the two Special Function Registers (SFRs), SCON and SBUF. SCON is the control register and SBUF is the data register. On reset, both SCON and SBUF return 00_H. The serial port control and status register is the SFR SCON. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8) and the serial port interrupt bits (TI and RI).

SBUF is the receive and transmit buffer of the serial interface. Writing to SBUF loads the transmit register and initiates transmission. This register is used for both transmit and receive data. Transmit data is written to this location and receive data is read from this location, but the two paths are independent.

Reading out SBUF accesses a physically separate receive register.

SBUF

Serial Data Buffer

(99_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
VAL							
rwh							

Field	Bits	Type	Description
VAL	[7:0]	rwh	Serial Interface Buffer Register

SCON

Serial Channel Control Register

(98_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI
rw	rw	rw	rw	rw	rwh	rwh	rwh

Field	Bits	Type	Description
RI	0	rwh	Receive Interrupt Flag This is set by hardware at the end of the 8th bit on mode 0, or at the half point of the stop bit in modes 1, 2, and 3. Must be cleared by software.

UART

Field	Bits	Type	Description
TI	1	rwh	Transmit Interrupt Flag This is set by hardware at the end of the 8th bit in mode 0, or at the beginning of the stop bit in modes 1, 2, and 3. Must be cleared by software.
RB8	2	rwh	Serial Port Receiver Bit 9 In modes 2 and 3, this is the 9th data bit received. In mode 1, this is the stop bit received. In mode 0, this bit is not used.
TB8	3	rw	Serial Port Transmitter Bit 9 In modes 2 and 3, this is the 9th data bit sent.
REN	4	rw	Enable Receiver of Serial Port 0 _B Serial reception is disabled. 1 _B Serial reception is enabled.
SM2	5	rw	Enable Serial Port Multiprocessor Communication in Modes 2 and 3 In mode 2 or 3, if SM2 is set to 1, RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 is set to 1, RI will not be activated if a valid stop bit (RB8) was not received. In mode 0, SM2 should be 0.
SM1	6	rw	Serial Port Operating Mode Selection See definition for SM0.
SM0	7	rw	Serial Port Operating Mode Selection [SM0, SM1] 00 _B Mode 0: 8-bit shift register, fixed baud rate ($f_{PCLK}/2$). 01 _B Mode 1: 8-bit UART, variable baud rate. 10 _B Mode 2: 9-bit UART, fixed baud rate ($f_{PCLK}/64$ or $f_{PCLK}/32$). 11 _B Mode 3: 9-bit UART, variable baud rate.

16.7.2 Baud-rate Generator Control and Status Registers

PCON

Power Control Register

(87_H)

Reset Value: 00_H

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
SMOD		0		GF1	GF0	0	IDLE
rw		r		rw	rw	r	rw

Field	Bits	Type	Description
SMOD	7	rw	Double Baud Rate Enable 0 _B Do not double the baud rate of serial interface in modes 1, 2 and 3. 1 _B Double the baud rate of serial interface in mode 2, and in modes 1 and 3 only if Timer 1 is used as variable baud rate source.
0	1,[6:4]	r	Reserved Returns 0 if read; should be written with 0.

BCON

Baud Rate Control Register

(F2_H)

Reset Value: 00_H

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
BGSEL		0	BRDIS		BRPRE		R
rw		r	rw		rw		rw

Field	Bits	Type	Description
R	0	rw	Baud Rate Generator Run Control Bit 0 _B Baud-rate generator disabled. 1 _B Baud-rate generator enabled. <i>Note: BR_VALUE should only be written if R = 0.</i>

Field	Bits	Type	Description
BRPRE	[3:1]	rw	Prescaler Bit Selects the input clock for f_{DIV} which is derived from the peripheral clock. $000_B f_{DIV} = f_{PCLK}$ $001_B f_{DIV} = f_{PCLK}/2$ $010_B f_{DIV} = f_{PCLK}/4$ $011_B f_{DIV} = f_{PCLK}/8$ $100_B f_{DIV} = f_{PCLK}/16$ $101_B f_{DIV} = f_{PCLK}/32$ Others: reserved
BRDIS	4	rw	Baud Rate Detection Disable 0_B Break/Synch detection is enabled. 1_B Break/Synch detection is disabled.
BGSEL	[7:6]	rw	Baud Rate Select for Detection For different values of BGSEL, the baud rate range for detection is defined by the following formula: $f_{pclk}/(2184 \cdot 2^{BGSEL}) < \text{baud rate range} < f_{pclk}/(72 \cdot 2^{BGSEL})$ where BGSEL = $00_B, 01_B, 10_B, 11_B$. See Table 16-6 for bit field BGSEL definition for different input frequencies.

LINST

LIN Status Register

(F5_H)

Reset Value: 00_H

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
BGS	SYNEN	ERRSYN	EOFSYN	BRK		0	
rw	rw	rwh	rwh	rwh		r	

Field	Bits	Type	Description
BRK	3	rwh	Break Field Flag This bit is set by hardware and can only be cleared by software. 0_B Break Field is not detected. 1_B Break Field is detected.

Field	Bits	Type	Description
EOFSYN	4	rwh	End of SYN Byte Interrupt Flag This bit is set by hardware and can only be cleared by software. 0 _B End of SYN Byte is not detected. 1 _B End of SYN Byte is detected.
ERRSYN	5	rwh	SYN Byte Error Interrupt Flag This bit is set by hardware and can only be cleared by software. 0 _B Error is not detected in SYN Byte. 1 _B Error is detected in SYN Byte.
SYNEN	6	rw	End of SYN Byte and SYN Byte Error Interrupts Enable 0 _B End of SYN Byte and SYN Byte Error Interrupts are not enabled. 1 _B End of SYN Byte and SYN Byte Error Interrupts are enabled.
BGS	7	rw	Baud Rate Generator Select 0 _B Baud-rate generator is selected. 1 _B Timer 1 is selected as baud rate generator.

16.7.3 Baud-rate Generator Timer/Reload Registers

The low and high bytes of the baud rate timer/reload register BG contains the 11-bit reload value for the baud rate timer and the 5-bit fractional divider selection.

Reading the low byte of register BG returns the content of the lower three bits of the baud rate timer and the FD_SEL setting, while reading the high byte returns the content of the upper 8 bits of the baud rate timer.

Writing to register BG loads the baud rate timer with the reload and fractional divider values from the BG register, the first instruction cycle after BCON.R is set.

BG should only be written if R = 0.

BGL

Baud Rate Timer/Reload Register, Low Byte
(F3_H)

Reset Value: 00_H

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
BR_VALUE			FD_SEL				
rwh			rw				

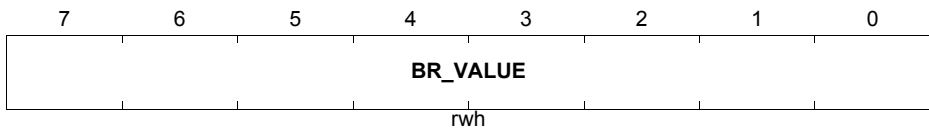
Field	Bits	Type	Description
FD_SEL	[4:0]	rw	Fractional Divider Selection Selects the fractional divider to be $n/32$, where n is the value of FD_SEL and is in the range of 0 to 31. For example, writing 0001_B to FD_SEL selects the fractional divider to be $1/32$. <i>Note: Fractional divider has no effect if BR_VALUE = 000_H.</i>
BR_VALUE	[7:0]	rwh	Baud Rate Timer/Reload Value The lower three bits of the 11-bit Baud Rate Timer/Reload value. See also description in BGH register.

BGH

Baud Rate Timer/Reload Register, High Byte (F4_H)

Reset Value: 00_H

RMAP: 0, PAGE: 5



Field	Bits	Type	Description
BR_VALUE	[7:0]	rwh	Baud Rate Timer/Reload Value The upper 8 bits of the 11-bit Baud Rate Timer/Reload value. When the 11-bit BR_VALUE is 000_H , baud-rate timer is bypassed.

17 Inter-IC Bus

17.1 Overview

The IIC provides an interface between the microcontroller and an IIC bus that conforms to the IIC Bus Protocol.

The IIC can operate in either master or slave mode. It performs arbitration in master mode to allow it to operate in multi-master systems. The IIC provides communication at data rates of up to 400 KBAud and features 7-bit addressing as well as 10-bit addressing.

Features:

- Operates in Master or slave mode
- Supports multi-master systems
- Performs arbitration and clock synchronization
- Supports 7-bit and 10-bit addressing modes
- Selectable baud rate generation of up to 400 KBAud (fast mode)
- Flexible control via interrupt service routines or polling

17.2 System Information

This section provides system information relevant to the IIC.

17.2.1 Pinning

The IIC pin assignment for XC82x is shown in [Table 17-1](#).

Table 17-1 IIC Pin Functions in XC82x

Pin	Function	Description	Selected By
P0.4	SCL_0	IIC Clock Input/Output	P0_ALTSEL0.P4 = 0 _B P0_ALTSEL1.P4 = 0 _B P0_ALTSEL2.P4 = 1 _B
P0.2	SCL_1	IIC Clock Input/Output	P0_ALTSEL0.P2 = 0 _B P0_ALTSEL1.P2 = 1 _B
P0.6	SDA_0	IIC Data Input/Output	P0_ALTSEL0.P6 = 0 _B P0_ALTSEL1.P6 = 0 _B P0_ALTSEL2.P6 = 1 _B
P0.3	SDA_1	IIC Data Input/Output	P0_ALTSEL0.P3 = 0 _B P0_ALTSEL1.P3 = 1 _B

Inter-IC Bus

Since SCL and SDA have input and output functions within the same set of pin selected, the input and output signal selection is done using just the Px_ALTSELn signals from the GPIO module. No separate input selection control (PISEL) is required..

If more than 1 set of input/output are selected via Px_ALTSELn register, the set with the lower set number has the highest priority. For example, if output signal SDA_0 and SDA_1 are selected at the same time, the input will only come from SDA_0.

17.2.1.1 Output Pin Configuration

The pin drivers that are assigned to the IIC lines provide open drain outputs. This ensures that the IIC does not put any load on the IIC bus lines while the IIC is not powered. Therefore, it is necessary to have external pull-up resistors (approximately 10 K Ω for operation at 100 KBaud, 2 K Ω for operation at 400 KBaud) on the IIC bus lines.

All pins of the IIC that are to be used for IIC bus communication must be switched to output and their alternate function must be enabled, before any communication can be established.

If not driven by the IIC module, the pins will switch off their drivers (i.e. driving 1 to an open drain output). Due to the external pull-up devices, the respective bus levels will then be 1, which is idle.

17.2.2 Clocking Configuration

The IIC runs on the PCLK at a frequency of either 8 MHz or 24 MHz.

If the IIC functionality is not required at all, it can be completely disabled by gating off its clock input for maximal power reduction. This is done by setting bit IIC_DIS in register PMCON1 as described below.

The bit field PAGE of SCU_PAGE register must be programmed before accessing the PMCON1 register.

PMCON1

Peripheral Management Control Register 1(EF_H)

Reset Value: DF_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	0	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	r	rw	rw	rw	rw	rw

Field	Bits	Type	Description
IIC_DIS	7	rw	IIC Disable Request. Active high. 0 IIC is in normal operation. 1 Request to disable the IIC. (default)
0	5	r	Reserved Returns 0 if read; should be written with 0.

17.2.3 Interrupt Events and Assignment

Table 17-2 lists the interrupt event source from the IIC, and the corresponding event interrupt enable bit and flag bit.

Table 17-2 IIC Interrupt Events

Event	Event Interrupt Enable Bit	Event Flag Bit
IIC interrupt	CNTR.IEN	CNTR.IFLG

Table 17-3 shows the interrupt node assignment for the IIC interrupt source.

Table 17-3 IIC Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
IIC interrupt	IEN1.EX2	—	43 _H

17.3 Status Code

The state of the IIC is defined by the 5-bit status code in STAT register.

When STAT contains the status code F8_H, no relevant status information is available, no interrupt is generated and the IFLG bit in the CNTR register is not set. All other status codes correspond to a defined state of the IIC. When any of these states is entered, the corresponding status code appears in this register and the IFLG bit in the CNTR register is set. When the IFLG bit is cleared, the status code returns to F8_H.

Table 17-4 Status Code

Value of STAT register	Status
00 _H	Bus error
08 _H	START condition transmitted
10 _H	Repeated START condition transmitted
18 _H	Address and write bit transmitted, ACK received
20 _H	Address and write bit transmitted, ACK not received
28 _H	Data byte transmitted in master mode, ACK received
30 _H	Data byte transmitted in master mode, ACK not received
38 _H	Arbitration lost in address or data byte
40 _H	Address and read bit transmitted, ACK received
48 _H	Address and read bit transmitted, ACK not received
50 _H	Data byte received in master mode, ACK transmitted
58 _H	Data byte received in master mode, ACK not transmitted
60 _H	Slave address and write bit received, ACK transmitted
68 _H	Arbitration lost in address as master, slave address and write bit received, ACK transmitted
70 _H	General call address received, ACK transmitted
78 _H	Arbitration lost in address as master, general call address received, ACK transmitted
80 _H	Data byte received after slave address received, ACK transmitted
88 _H	Data byte received after slave address received, ACK not transmitted
90 _H	Data byte received after general call address received, ACK transmitted
98 _H	Data byte received after general call address received, ACK not transmitted

Table 17-4 Status Code

Value of STAT register	Status
A0 _H	STOP or repeated START mode received in slave mode
A8 _H	Slave address and read bit received, ACK transmitted
B0 _H	Arbitration lost in address as master, slave address and read bit received, ACK transmitted
B8 _H	Data byte transmitted in slave mode, ACK received
C0 _H	Data byte transmitted in slave mode, ACK not received
C8 _H	Last byte transmitted in slave mode, ACK received
D0 _H	Second address byte and write bit transmitted, ACK received
D8 _H	Second address byte and write bit transmitted, ACK not received
E0 _H	Unused
E8 _H	Unused
F0 _H	Unused
F8 _H	No relevant status information, IFLG = 0

If an illegal condition occurs on the IIC bus, the bus error state is entered (status code 00_H). To recover from this state, the STP bit in the CNTR register must be set and the IFLG bit cleared. The IIC will then return to idle state (status code F8_H): no STOP condition will be transmitted on the IIC bus.

To request resumption of transmission, set the STA bit to 1 at the same time as the STP bit is set. The IIC will then send a START on recovery from the bus error.

17.4 Baud Rate Generation

To support a wide range of input clock frequencies (f_{IIC}), two dividers are implemented to generate the required baud rate on the IIC bus in master mode. The two dividers are defined by the PREDIV and BRP bits in BRCR register.

The baud rate is calculated by the formulae:

(17.1)

$$f_{\text{OSCL}} = \frac{f_{\text{PCLK}}}{2^{\text{PREDIV}} \times (\text{BRP} + 1) \times 10}$$

Table 17-5 shows various commonly used baud rates together with the required dividers setting.

Table 17-5 Baud Rate Selection

f_{PCLK}	Baud Rate = 100 Kbaud		Baud Rate = 400 Kbaud	
	PREDIV	BRP+1	PREDIV	BRP+1
8 MHz	1 (1 _H)	4 (4 _H)	1 (1 _H)	1 (1 _H)
24 MHz	1 (1 _H)	12 (C _H)	1 (1 _H)	3 (3 _H)

The frequency at which the IIC bus is sampled is calculated by the formulae:

(17.2)

$$f_{SAMP} = \frac{f_{PCLK}}{2^{PREDIV}}$$

To ensure correct detection of START and STOP conditions on the bus, the IIC must sample the IIC bus at least ten times faster than the bus clock speed of the fastest master on the bus. The sampling frequency should therefore be at least 1 MHz in standard mode or 4 MHz in fast mode to guarantee correct operation with other bus masters.

17.5 Clock Synchronization

If another device on the IIC bus drives the clock line when the IIC is in master mode, the IIC will synchronize its clock to the IIC bus clock. The high period of the clock will be determined by the device that generates the shortest high clock period. The low period of the clock will be determined by the device that generates the longest low clock period.

When the IIC is in master mode and is communicating with a slow slave, the slave may stretch each bit period by holding the SCL line low until it is ready for the next bit. The IIC will automatically re-synchronize as described above.

When the IIC is in slave mode, it will hold the SCL line low after each byte has been transferred until IFLG has been cleared in the CNTR register.

17.6 Bus Arbitration

In master mode, the IIC will check that each transmitted logic 1 appears on the IIC bus as a logic 1. If another device on the bus over-rides and pulls the SDA line low, arbitration is lost. If arbitration is lost during the transmission of a data byte or a Not-Acknowledge bit, the IIC will return to idle state. If arbitration is lost during the transmission of an address, the IIC will switch to slave mode so that it can recognize its own slave address or the general call address.

17.7 Software Reset

A software reset may be applied to the IIC module by writing any value to register SRST (address DF_H). This sets the IIC back to idle (STAT set to F8_H) and sets the STP, STA and IFLG bits of the CNTR register to 0.

17.8 Operating Modes

All operating modes of the IIC module require the ENAB bit of register CNTR to be set to 1.

17.8.1 Master Transmit

In the master transmit mode, the IIC transmits a number of bytes to a slave receiver.

The master transmit mode is entered by setting the STA bit in register CNTR to '1'. The IIC will then test the IIC bus and will transmit a START condition when the bus is free. When a START condition has been transmitted, the IFLG bit will be set and the status code in the STAT register will be 08_H. Before this interrupt is serviced, the DATA register must be loaded with either a 7-bit slave address or the first part of a 10-bit slave address, with the LSB cleared to '0' (i.e. with an added Write bit) to specify transmit mode. The IFLG bit should now be cleared to '0' to prompt the transfer to continue.

After the 7-bit slave address (or the first part of a 10-bit address) plus the Write bit have been transmitted, IFLG will be set again. A number of status codes are possible in the STAT register:

Table 17-6 Status Code after Address is Transmitted in Master Transmit Mode

Code	IIC State	CPU Response	Next IIC Action
18 _H	Addr + W, ACK received	For 7-bit address: Write byte to DATA, clear IFLG Or Set STA, clear IFLG Or Set STP, clear IFLG Or Set STA and STP, clear IFLG For 10-bit address: Write extended address byte to DATA, clear IFLG	Transmit data byte, receives ACK Transmit repeated START Transmit STOP Transmit STOP then START Transmit extended data byte
20 _H	Addr + W, ACK not received	Same as for code 18 _H	Same as for code 18 _H
38 _H	Arbitration lost	Clear IFLG Or set STA, clear IFLG	Return to idle state Transmit START when bus is free
68 _H	Arbitration lost, SLA + W received, ACK transmitted	Clear IFLG, AAK = 0 Or clear IFLG, AAK = 1	Receive data byte, transmit not ACK Receive data byte, transmit ACK
78 _H	Arbitration lost, general call address received, ACK transmitted	Same as for code 68 _H	Same as for code 68 _H
B0 _H	Arbitration lost, SLA + R received, ACK transmitted	Write byte to DATA, clear IFLG, AAK = 0 Or write byte to DATA, clear IFLG, AAK = 1	Transmit last byte, receive ACK Transmit data byte, receive ACK

Inter-IC Bus

If 10-bit addressing is being used, then after the first part of a 10-bit address plus the Write bit have been successfully transmitted, the status code will be 18_H or 20_H.

After this interrupt has been serviced and the second part of the address transmitted, the STAT register will contain one of the following codes:

Table 17-7 Status Code after Second Address Byte (for 10-bit Addressing) is Transmitted in Master Transmit Mode

Code	IIC State	CPU Response	Next IIC Action
38 _H	Arbitration lost	Clear IFLG Or set STA, clear IFLG	Return to idle state Transmit START when bus is free
68 _H	Arbitration lost, SLA + W received, ACK transmitted	Clear IFLG, AAK = 0 Or clear IFLG, AAK = 1	Receive data byte, transmit not ACK Receive data byte, transmit ACK
B0 _H	Arbitration lost, SLA + R received, ACK transmitted	Write byte to DATA, clear IFLG, AAK = 0 Or write byte to DATA, clear IFLG, AAK = 1	Transmit last byte, receive ACK Transmit data byte, receive ACK
D0 _H	Second address byte + W transmitted, ACK received	Write byte to DATA, clear IFLG Or set STA, clear IFLG Or set STP, clear IFLG Or set STA and STP, clear IFLG	Transmit data byte, receives ACK Transmit repeated START Transmit STOP Transmit STOP then START
D8 _H	Second address byte + W transmitted, ACK not received	Same as for code D0 _H	Same as for code D0 _H

If a repeated START condition has been transmitted, the status code will be 10_H instead of 08_H.

After each data byte has been transmitted, IFLG will be set and one of three status codes will be in the STAT register:

Table 17-8 Status Code after Data is Transmitted in Master Transmit Mode

Code	IIC State	CPU Response	Next IIC Action
28 _H	Data byte transmitted, ACK received	Write byte to DATA, clear IFLG	Transmit data byte, receives ACK
		Or set STA, clear IFLG	Transmit repeated START
		Or set STP, clear IFLG	Transmit STOP
		Or set STA and STP, clear IFLG	Transmit STOP then START
30 _H	Data byte transmitted, ACK not received	Same as for code 28 _H	Same as for code 28 _H
38 _H	Arbitration lost	Clear IFLG	Return to idle state
		Or set STA, clear IFLG	Transmit START when bus is free

When all bytes have been transmitted, the STP bit should be set by writing a 1 to this bit in the CNTR register. The IIC will then transmit a STOP condition, clear the STP bit and return to idle state (status code F8_H).

17.8.2 Master Receive

In the master receive mode, the IIC will receive a number of bytes from a slave transmitter.

After the START condition has been transmitted, the IFLG bit will be set and status code 08_H will be in the STAT register. The DATA register should now be loaded with the slave address (or the first part of a 10-bit slave address), with the LSB set to '1' to signify a Read. The IFLG bit should now be cleared to '0' to prompt the transfer to continue.

When the 7-bit slave address (or the first part of a 10-bit address) and the Read bit have been transmitted, the IFLG bit will be set again. A number of status codes are possible in the STAT register:

Table 17-9 Status Code after Address is Transmitted in Master Receive Mode

Code	IIC State	CPU Response	Next IIC Action
40 _H	Addr + R transmitted, ACK received	Clear IFLG, AAK = 0 Or clear IFLG, AAK = 1	Receive data byte, transmit not ACK Receive data byte, transmit ACK
48 _H	Addr + R transmitted, ACK not received	Set STA, clear IFLG Or set STP, clear IFLG Or set STA and STP, clear IFLG	Transmit repeated START Transmit STOP Transmit STOP then START
38 _H	Arbitration lost	Clear IFLG Or set STA, clear IFLG	Return to idle state Transmit START when bus is free
68 _H	Arbitration lost, SLA + W received, ACK transmitted	Clear IFLG, AAK = 0 Or clear IFLG, AAK = 1	Receive data byte, transmit not ACK Receive data byte, transmit ACK
78 _H	Arbitration lost, general call address received, ACK transmitted	Same as for code 68 _H	Same as for code 68 _H
B0 _H	Arbitration lost, SLA + R received, ACK transmitted	Write byte to DATA, clear IFLG, AAK = 0 Or write byte to DATA, clear IFLG, AAK = 1	Transmit last byte, receive ACK Transmit data byte, receive ACK

If 10-bit addressing is being used, the slave is first addressed using the full 10-bit address plus the Write bit. The master then issues a restart followed the first part of the 10-bit address again, but plus the Read bit – after which the status code will be 40_H or 48_H. It is the responsibility of the slave to remember that it had been selected prior to the restart.

If a repeated START condition has been transmitted, the status code will be 10_H instead of 08_H.

Inter-IC Bus

After each data byte has been received, IFLG will be set and one of three status codes will be in the STAT register:

Table 17-10 Status Code after Data is Received in Master Receive Mode

Code	IIC State	CPU Response	Next IIC Action
50 _H	Data byte received, ACK transmitted	Read DATA, clear IFLG, AAK = 0	Receive data byte, transmit not ACK
		Or read DATA, clear IFLG, AAK = 1	Receive data byte, transmit ACK
58 _H	Data byte received, not ACK transmitted	Read DATA, set STA, clear IFLG	Transmit repeated START
		Or read DATA, set STP, clear IFLG	Transmit STOP
		Or read DATA, set STA and STP, clear IFLG	Transmit STOP then START
38 _H	Arbitration lost in not ACK bit	Clear IFLG	Return to idle state
		Or set STA, clear IFLG	Transmit START when bus is free

When all bytes have been received, a not ACK should be transmitted then the STP bit should be set by writing a '1' to this bit in the CNTR register. The IIC will transmit a STOP condition, clear the STP bit and return to idle state (status code F8_H).

17.8.3 Slave Transmit

In the slave transmit mode, a number of bytes are transmitted to a master receiver. For the IIC to respond, the AAK bit in the CNTR register needs to be set.

The IIC will enter slave transmit mode when it receives its own slave address and a Read bit after a START condition. The IIC will then transmit an acknowledge bit and set the IFLG bit in the CNTR register. The STAT register will contain the status code A8_H.

Note: Where the IIC has an extended slave address (signified by 11110_B in ADDR[7:3]), it will first be selected, then there will be a restart followed by another address byte. If this address byte matches the value stored in ADDR, the IIC will transmit an acknowledge after this address byte is received. An interrupt will be generated, IFLG will be set and the status will be A8_H. No second address byte will be sent by the master: it is up to the slave to remember that it had been selected prior to the restart.

Inter-IC Bus

Slave transmit mode can also be entered directly from a master mode if arbitration is lost in master mode during the transmission of an address and the slave address and Read bit are received. The status code in the STAT register will then be B0_H.

The data byte to be transmitted should then be loaded into the DATA register and IFLG cleared. When the IIC has transmitted the byte and received an acknowledge, IFLG will be set and the STAT register will contain B8_H. Once the last byte to be transmitted has been loaded into the DATA register, the AAK bit should be cleared when IFLG is cleared. After the last byte has been transmitted, IFLG will be set and the STAT register will contain C8_H. The IIC will then return to idle state (status code F8_H). The AAK bit must be set to 1 before slave mode can be entered again.

If no acknowledge is received after transmitting a byte, IFLG will be set and the STAT register will contain C0_H. The IIC will then return to idle state.

If the STOP condition is detected after an acknowledge bit, the IIC will return to idle state.

17.8.4 Slave Receive

In the slave receive mode, a number of data bytes are received from a master transmitter.

The IIC will enter slave receive mode when it receives its own slave address and a Write bit (LSB=0) after a START condition. The IIC will then transmit an acknowledge bit and set the IFLG bit in the CNTR register: the STAT register will then contain status code 60_H. The IIC will also enter slave receive mode when it receives the general call address 00_H (if the GCE bit in the ADDR register is set). The status code will then be 70_H.

Note: Where the IIC has an extended slave address (signified by 1110_B in ADDR[7:3]), it will transmit an acknowledge after the first address byte is received but no interrupt will be generated, IFLG will not be set and the status will not change. Only after the second address byte has been received will the IIC generate an interrupt, set the IFLG bit and the status code as described above.

Slave receive mode can also be entered directly from a master mode if arbitration is lost in master mode during the transmission of an address and the slave address and Write bit (or the general call address if bit GCE in the ADDR register is set to 1) are received. The status code in the STAT register will then be 68_H if the slave address was received or 78_H if the general call address was received. The IFLG bit must be cleared to '0' to allow the data transfer to continue.

If the AAK bit in the CNTR register is set to 1, then after each byte is received, an acknowledge bit (low level on SDA) is transmitted and the IFLG bit is set: the STAT register will then contain status code 80_H (or 90_H if slave receive mode was entered with the general call address). The received data byte can be read from the DATA register and the IFLG bit must be cleared to allow the transfer to continue. When the STOP condition or a repeated START condition is detected after the acknowledge bit, then the IFLG bit is set and the STAT register will contain status code A0_H.

Inter-IC Bus

If the AAK bit is cleared to 0 during a transfer, the IIC will transmit a not acknowledge bit (high level on SDA) after the next byte is received, and set the IFLG bit. The STAT register will contain status code 88_H (or 98_H if slave receive mode was entered with the general call address). When the IFLG bit has been cleared to 0, the IIC will return to idle state (status code $F8_H$).

17.9 Registers Description

The IIC Special Function Registers are accessed from the standard (non-mapped) SFR area. [Table 17-11](#) lists the IIC registers with their addresses.

Table 17-11 Register Map

Address	Register	Description
DA _H	IIC_ADDR	Slave Address Register
DB _H	IIC_DATA	Data Byte Register
DC _H	IIC_CNTR	Control Register
DD _H	IIC_STAT	Status Register (read only)
DD _H	IIC_BRCCR	Clock Control Register (write only)
DE _H	IIC_ADDRX	Extended Slave Address Register
DF _H	IIC_SRST	Software Reset Register

17.9.1 Slave Address Registers

The ADDR register contains the bit to enable the general call address option and the device address of the IIC in 7-bit addressing mode. For 10-bit addressing, part of the address bits is located in ADDR_X register.

IIC_ADDR

Slave Address Register

(DA_H)

Reset Value: 00_H

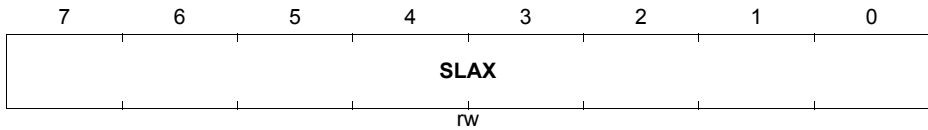
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
SLA							GCE
rw							rw

Field	Bits	Type	Description
GCE	0	rw	General Call Address Enable 0 _B General call address option is disabled. 1 _B General call address option is enabled.
SLA	[7:1]	rw	Slave Address For 7-bit addressing, SLA contains the 7-bit address of the IIC when in slave mode. For 10-bit addressing, the uppermost five bits SLA[6:2] are fixed at 11110 _B , while SLA[1:0] contains the uppermost two bits of the 10-bit slave address. The remaining lower 8 bits are located in IIC_ADDRX register.

Inter-IC Bus

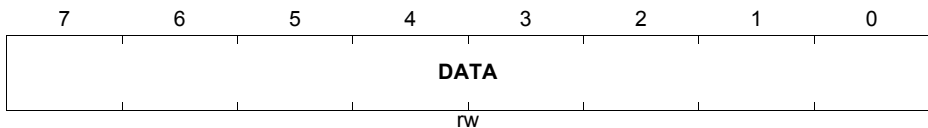
The ADDR_X register contains the lower 8-bit slave address of the device in 10-bit addressing mode.

IIC_ADDRX
Extended Slave Address Register (DE_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


Field	Bits	Type	Description
SLAX	[7:0]	rw	Extended Slave Address For 10-bit addressing, SLAX contains the lower 8-bit address of the IIC when in slave mode.

17.9.2 Data Register

The DATA register contains the data byte or slave address to be transmitted or the data byte that has just been received.

IIC_DATA
Data Register (DB_H)
Reset Value: 00_H
RMAP: 0, PAGE: X


Field	Bits	Type	Description
DATA	[7:0]	rw	Data Byte Data to be sent or received.

17.9.3 Control Register

The CNTR register is used to configure the IIC and generate START and STOP conditions. It also contains the interrupt status flag.

IIC_CNTR

Control Register

(DC_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
IEN	ENAB	STA	STP	IFLG	AAK	0	
rw	rw	rwh	rwh	rwh	rw	r	

Field	Bits	Type	Description
AAK	2	rw	Assert Acknowledge 0 _B A NACK is sent when a data byte is received in master or slave mode. 1 _B An ACK is sent if one of the following has been received: 1) a matching 7-bit or either byte of the 10-bit slave address; 2) general call address; 3) a data byte in master or slave mode. <i>Note: If the AAK bit is cleared to 0 in slave transmitter mode, the byte in the DATA register is assumed to be the last byte.</i>
IFLG	3	rwh	Interrupt Flag The IFLG bit is set by hardware and can only be cleared by software. 0 _B No interrupt event has occurred. 1 _B An interrupt event has occurred.

Inter-IC Bus

Field	Bits	Type	Description
STP	4	rwh	<p>Master Mode Stop</p> <p>When the STP bit is set to 1, the IIC will transmit a STOP condition.</p> <p>If the STP bit is set while the IIC is in slave mode, no STOP condition will be transmitted on the IIC bus but the IIC behaves as if a STOP condition has been received.</p> <p>If both STA and STP bits are set, the IIC will first transmit a STOP condition (if in master mode) before transmitting a START condition.</p> <p>0_B The IIC does not transmit any STOP condition. 1_B The IIC transmits a STOP condition on the IIC bus.</p> <p><i>Note: The STP bit is cleared automatically after the STOP condition has been sent ; writing 0 to this bit has no effect.</i></p>
STA	5	rwh	<p>Master Mode Start</p> <p>When the STA bit is set to 1, the IIC will enter master mode and transmit a START condition once the IIC bus is free.</p> <p>If the IIC is already in master mode and one or more bytes have been transmitted, a repeated START condition will be transmitted.</p> <p>If the IIC is still being accessed in slave mode, the IIC will complete the data transfer in slave mode and enter master mode once the IIC bus has been released.</p> <p>0_B The IIC does not enter master mode. 1_B The IIC enters master mode and transmits a START condition on the IIC bus.</p> <p><i>Note: The STA bit is cleared automatically after the START condition has been sent ; writing 0 to this bit has no effect.</i></p>
ENAB	6	rw	<p>IIC Enable</p> <p>0_B The IIC is disabled. 1_B The IIC is enabled.</p>
IEN	7	rw	<p>Interrupt Enable</p> <p>0_B The interrupt is disabled. 1_B The interrupt is enabled.</p>

Inter-IC Bus

Field	Bits	Type	Description
0	[1:0]	r	Reserved Returns 0 if read; should be written with 0.

17.9.4 Status Register

The read-only STAT register stores the 5-bit status code of the IIC.

IIC_STAT

Status Register [Read Mode]

(DD_H)

Reset Value: F8_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
STAT					0		
r					r		

Field	Bits	Type	Description
STAT	[7:3]	r	Status Code 5-bit status code (see Table 17-4).
0	[2:0]	r	Reserved Returns 0 if read; should be written with 0.

17.9.5 Baud Rate Control Register

The write-only BRCCR register controls the sampling frequency of the IIC and the baud rate of the IIC in master mode.

IIC_BRCCR

Baud Rate Control Register [Write Mode]

(DD_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
0	BRP				PREDIV		
W	W				W		

Field	Bits	Type	Description
PREDIV	[2:0]	w	Predivider for Baud Rate Generation 000 _B Predivider factor of 1 is used. 001 _B Predivider factor of 2 is used. 010 _B Predivider factor of 4 is used. 011 _B Predivider factor of 8 is used. 100 _B Predivider factor of 16 is used. 101 _B Predivider factor of 32 is used. 110 _B Predivider factor of 64 is used. 111 _B Predivider factor of 128 is used.
BRP	[6:3]	w	Baud Rate Prescaler Determines the baud rate for the IIC in master mode.
0	7	w	Reserved Should be written with 0.

17.9.6 Software Reset Register

The SRST register provides for a software reset on the IIC.

IIC_SRST

Software Reset Register

(DF_H)

Reset Value: XX_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
SRST							
W							

Field	Bits	Type	Description
SRST	[7:0]	w	Software Reset Writing any value to the SRST bit field triggers a soft reset on the IIC.

18 High-Speed Synchronous Serial Interface

18.1 Overview

The High-Speed Synchronous Serial Interface (SSC) supports both full-duplex and half-duplex serial synchronous communication. The serial clock signal can be generated by the SSC itself (Master Mode) through its own 16-bit baud rate generator, or can be received from an external master (Slave Mode). Data width, shift direction, clock polarity, and phase are programmable. This allows communication with SPI-compatible devices using other synchronous serial interfaces. Transmission and reception of data is double-buffered.

Features

- Master and Slave Mode operation
 - Full-duplex or half-duplex operation
- Transmit and receive buffered
- Flexible data format
 - Programmable number of data bits: 2 to 8 bits
 - Programmable shift direction: Least Significant Bit (LSB) or Most Significant Bit (MSB) shift first
 - Programmable clock polarity: idle low or high state for the shift clock
 - Programmable clock/data phase: data shift with leading or trailing edge of the shift clock
- Variable baud rate
- Compatible with Serial Peripheral Interface (SPI)
- Interrupt generation
 - On a transmitter empty condition
 - On a receiver full condition
 - On an error condition (receive, phase, baud rate, transmit error)

Figure 18-1 shows the block diagram of the SSC.

High-Speed Synchronous Serial Interface

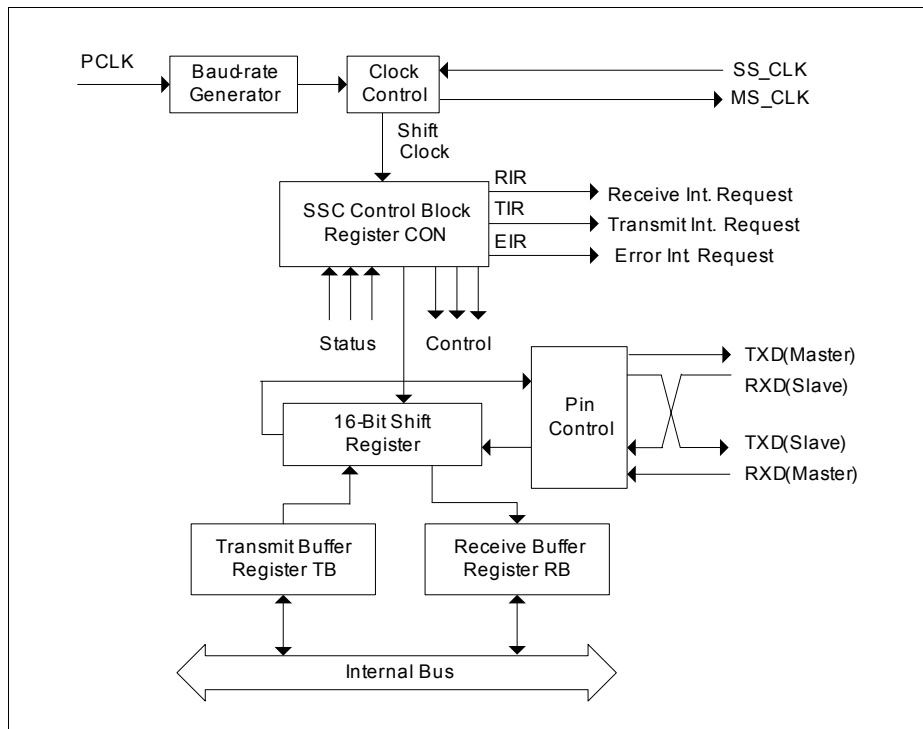


Figure 18-1 SSC Block Diagram

18.2 System Information

This section provides system information relevant to the SSC.

18.2.1 Pinning

The SSC pin assignment for XC82x is shown in [Table 18-1](#).

High-Speed Synchronous Serial Interface
Table 18-1 SSC Pin Functions in XC82x

Pin	Function	Description	Selected By
P0.4	SCK_0	SSC Clock Input/Output	For Input: MODPISEL.CIS = 0 _B For Output: P0_ALTSEL0.P4 = 0 _B P0_ALTSEL1.P4 = 1 _B P0_ALTSEL2.P4 = 0 _B
P2.2	SCK_1	SSC Clock Input	For Input: MODPISEL.CIS = 1 _B
P0.5	MTSR_0	SSC Master Transmit Output /Slave Receive Input	For Input: MODPISEL.SIS = 000 _B For Output: P0_ALTSEL0.P5 = 0 _B P0_ALTSEL1.P5 = 1 _B P0_ALTSEL2.P5 = 0 _B
P0.6	MTSR_1	SSC Slave Receive Input	For Input: MODPISEL.SIS = 001 _B
P0.0	MTSR_2	SSC Master Transmit Output /Slave Receive Input	For Input: MODPISEL.SIS = 010 _B For Output: P0_ALTSEL0.P0 = 0 _B P0_ALTSEL1.P0 = 1 _B
P0.1	MTSR_3	SSC Slave Receive Input	For Input: MODPISEL.SIS = 011 _B
P2.1	MTSR_4	Slave Receive Input	For Input: MODPISEL.SIS = 100 _B
P0.6	MRST_0	SSC Master Receive Input /Slave Transmit Output	For Input: MODPISEL.MIS = 00 _B For Output: P0_ALTSEL0.P6 = 0 _B P0_ALTSEL1.P6 = 1 _B P0_ALTSEL2.P6 = 0 _B
P0.5	MRST_1	SSC Master Receive Input	For Input: MODPISEL.MIS = 01 _B

High-Speed Synchronous Serial Interface

Table 18-1 SSC Pin Functions in XC82x

Pin	Function	Description	Selected By
P0.1	MRST_2	SSC Master Receive Input /Slave Transmit Output	For Input: MODPISEL.MIS = 10 _B For Output: P0_ALTSEL0.P1 = 0 _B P0_ALTSEL1.P1 = 1 _B
P0.0	MRST_3	SSC Master Receive Input	For Input: MODPISEL.MIS = 11 _B

The SSC uses three lines to communicate with the external world. Pin SCK serves as the clock line, while pins MRST (Master Receive/Slave Transmit) and MTSR (Master Transmit/Slave Receive) serve as the serial data input/output lines.

Each of the input lines can be selected from one of several different sources. The selection is done in the MODPISEL register. Similarly, each of the output lines can be also selected from several different sources. However, the selection is done at the respective Ports' ALTSELx registers.

The bit field PAGE of SCU_PAGE register must be programmed before accessing the MODPISEL register.

MODPISEL
Peripheral Input Select Register
(F3_H)
Reset Value: 00_H
RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	CIS		SIS		0		MIS
r	rw		rw		r		rw

Field	Bits	Type	Description
MIS	[1:0]	rw	Master Mode Receive Input Select 00 SSC Master Receiver Input 0 is selected. 01 SSC Master Receiver Input 1 is selected. 10 SSC Master Receiver Input 2 is selected. 11 SSC Master Receiver Input 3 is selected.

High-Speed Synchronous Serial Interface

Field	Bits	Type	Description
SIS	[5:3]	rw	Slave Mode Receive Input Select 000 SSC Slave Receiver Input 0 is selected. 001 SSC Slave Receiver Input 1 is selected. 010 SSC Slave Receiver Input 2 is selected. 011 SSC Slave Receiver Input 3 is selected. 100 SSC Slave Receiver Input 4 is selected. 101 Reserved. 110 Reserved. 111 Reserved.
CIS	6	rw	Slave Mode Clock Input Select 0 SSC Slave Clock Input 0 is selected. 1 SSC Slave Clock Input 1 is selected.
0	2, 7	r	Reserved Returns 0 if read; should be written with 0.

High-Speed Synchronous Serial Interface

18.2.2 Clocking Configuration

The SSC runs on the PCLK at a frequency of either 8 MHz or 24 MHz.

If the SSC functionality is not required at all, it can be completely disabled by gating off its clock input for maximal power reduction.

The bit field PAGE of register SCU_PAGE must be programmed before accessing the PMCON1 register.

PMCON1

Peripheral Management Control Register 1(EF_H)

Reset Value: DF_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	0	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	r	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SSC_DIS	1	rw	SSC Disable Request. Active high. 0 SSC is in normal operation. 1 Request to disable the SSC. (default)
0	5	r	Reserved Returns 0 if read; should be written with 0.

High-Speed Synchronous Serial Interface

18.2.3 Interrupt Events and Assignment

Table 18-2 lists the interrupt event sources from the SSC, and the interrupt node assignment for each SSC interrupt source.

Note: All SSC interrupt enable bits and flags are located at the node level.

Table 18-2 SSC Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
Start-of-Transmission	MODIEN.TIREN; IEN1.ESSC	IRCON1.TIR	3BH
End-of-Transmission	MODIEN.RIREN; IEN1.ESSC	IRCON1.RIR	3BH
Occurrence-of-Error	MODIEN.EIREN; IEN1.ESSC	IRCON1.EIR	3BH

The three interrupt events of SSC module are of interrupt structure 2 and register MODIEN is used to enable the interrupt. The bit field PAGE of register SCU_PAGE must be programmed before accessing the MODIEN register.

MODIEN

Peripheral Interrupt Enable Register (F7_H)

Reset Value: 07_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
CCU6SR3 EN	CCU6SR2 EN	0			RIREN	TIREN	EIREN
rw	rw	r			rw	rw	rw

Field	Bits	Type	Description
EIREN	0	rw	SSC Error Interrupt Enable 0 Error interrupt is disabled 1 Error interrupt is enabled
TIREN	1	rw	SSC Transmit Interrupt Enable 0 Transmit interrupt is disabled 1 Transmit interrupt is enabled
RIREN	2	rw	SSC Receive Interrupt Enable 0 Receive interrupt is disabled 1 Receive interrupt is enabled

High-Speed Synchronous Serial Interface

Field	Bits	Type	Description
0	[5:3]	r	Reserved Returns 0 if read; should be written with 0.

18.3 General Operation

The SSC supports full-duplex and half-duplex synchronous communication up to 12 MBaud (@ 24 MHz module clock). The serial clock signal can be generated by the SSC itself (Master Mode) or can be received from an external master (Slave Mode). Data width, shift direction, clock polarity, and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data is double-buffered. A 16-bit baud-rate generator provides the SSC with a separate serial clock signal.

The SSC can be configured in a very flexible way, so it can be used with other synchronous serial interfaces, can serve for master/slave or multimaster interconnections or can operate compatible with the popular SPI interface. Thus, the SSC can be used to communicate with shift registers (I/O expansion), peripherals (e.g. EEPROMs, etc.) or other controllers (networking). The SSC supports half-duplex and full-duplex communication. Data is transmitted or received on lines TXD and RXD, normally connected with pins MTSR (Master Transmit/Slave Receive) and MRST (Master Receive/Slave Transmit). The clock signal is output via line MS_CLK (Master Serial Shift Clock) or input via line SS_CLK (Slave Serial Shift Clock). Both lines are normally connected to pin SCLK.

18.3.1 Operating Mode Selection

The operating mode of the serial channel SSC is controlled by its control register CON. This register serves two purposes:

- During programming (SSC disabled by CON.EN = 0), it provides access to a set of control bits
- During operation (SSC enabled by CON.EN = 1), it provides access to a set of status flags.

The shift register of the SSC is connected to both the transmit lines and the receive lines via the pin control logic. Transmission and reception of serial data are synchronized and take place at the same time, i.e. the same number of transmitted bits is also received. Transmit data is written into the Transmit Buffer (TB) and is moved to the shift register as soon as this is empty. An SSC master (CON.MS = 1) immediately begins transmitting, while an SSC slave (CON.MS = 0) will wait for an active shift clock. When the transfer starts, the busy flag CON.BSY is set and the Transmit Interrupt Request line TIR will be activated to indicate that register TB may be reloaded again. When the programmed number of bits (2 ... 8) has been transferred, the contents of the shift register are moved to the Receive Buffer RB and the Receive Interrupt Request line RIR

High-Speed Synchronous Serial Interface

will be activated. If no further transfer is to take place (TB is empty), CON.BSY will be cleared at the same time. Software should not modify CON.BSY, as this flag is hardware controlled.

Note: The SSC starts transmission and sets CON.BSY minimum two clock cycles after transmit data is written into TB. Therefore, it is not recommended to poll CON.BSY to indicate the start and end of a single transmission. Instead, interrupt service routine should be used if interrupts are enabled, or the interrupt flags IRCON1.TIR and IRCON1.RIR should be polled if interrupts are disabled.

Note: Only one SSC (etc.) can be master at a given time.

The transfer of serial data bits can be programmed in many respects:

- The data width can be specified from 2 bits to 8 bits
- A transfer may start with either the LSB or the MSB
- The shift clock may be idle low or idle high
- The data bits may be shifted with the leading edge or the trailing edge of the shift clock signal
- The baud rate may be set from 183.11 Baud up to 12 MBaud (@ 24 MHz module clock)
- The shift clock can be generated (MS_CLK) or can be received (SS_CLK)

These features allow the adaptation of the SSC to a wide range of applications requiring serial data transfer.

The Data Width Selection supports the transfer of frames of any data length, from 2-bit “characters” up to 8-bit “characters”. Starting with the LSB (CON.HB = 0) allows communication with SSC devices in Synchronous Mode or with 8051 like serial interfaces for example. Starting with the MSB (CON.HB = 1) allows operation compatible with the SPI interface.

Regardless of the data width selected and whether the MSB or the LSB is transmitted first, the transfer data is always right-aligned in registers TB and RB, with the LSB of the transfer data in bit 0 of these registers. The data bits are rearranged for transfer by the internal shift register logic. The unselected bits of TB are ignored; the unselected bits of RB will not be valid and should be ignored by the receiver service routine.

The Clock Control allows the adaptation of transmit and receive behavior of the SSC to a variety of serial interfaces. A specific shift clock edge (rising or falling) is used to shift out transmit data, while the other shift clock edge is used to latch in receive data. Bit CON.PH selects the leading edge or the trailing edge for each function. Bit CON.PO selects the level of the shift clock line in the idle state. Thus, for an idle-high clock, the leading edge is a falling one, a 1-to-0 transition (see [Figure 18-2](#)).

High-Speed Synchronous Serial Interface

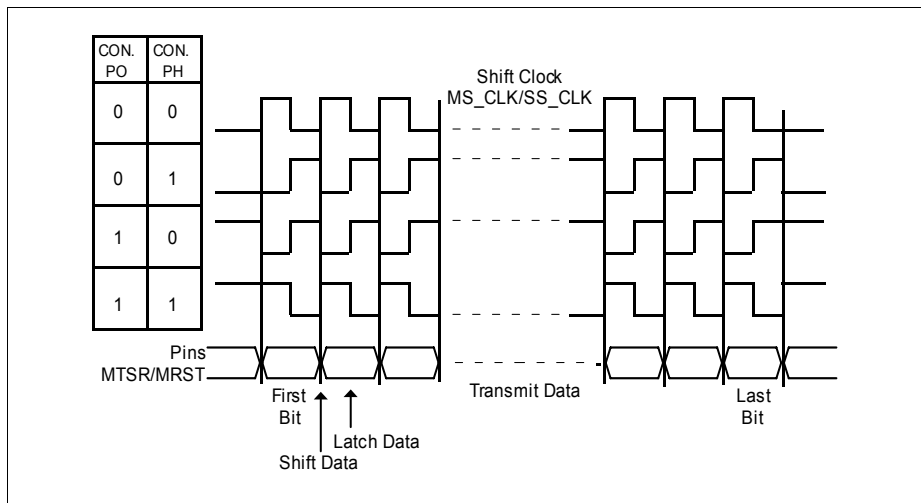


Figure 18-2 Serial Clock Phase and Polarity Options

18.3.2 Full-Duplex Operation

The various devices are connected through three lines. The definition of these lines is always determined by the master: the line connected to the master's data output line TXD is the transmit line; the receive line is connected to its data input line RXD; the shift clock line is either MS_CLK or SS_CLK. Only the device selected for master operation generates and outputs the shift clock on line MS_CLK. Since all slaves receive this clock, their pin SCLK must be switched to input mode. The output of the master's shift register is connected to the external transmit line, which in turn is connected to the slaves' shift register input. The output of the slaves' shift register is connected to the external receive line in order to enable the master to receive the data shifted out of the slave. The external connections are hard-wired, the function and direction of these pins is determined by the master or slave operation of the individual device.

Note: The shift direction shown in the figure applies for MSB-first operation as well as for LSB-first operation.

When initializing the devices in this configuration, one device must be selected for master operation while all other devices must be programmed for slave operation. Initialization includes the operating mode of the device's SSC and also the function of the respective port lines.

High-Speed Synchronous Serial Interface

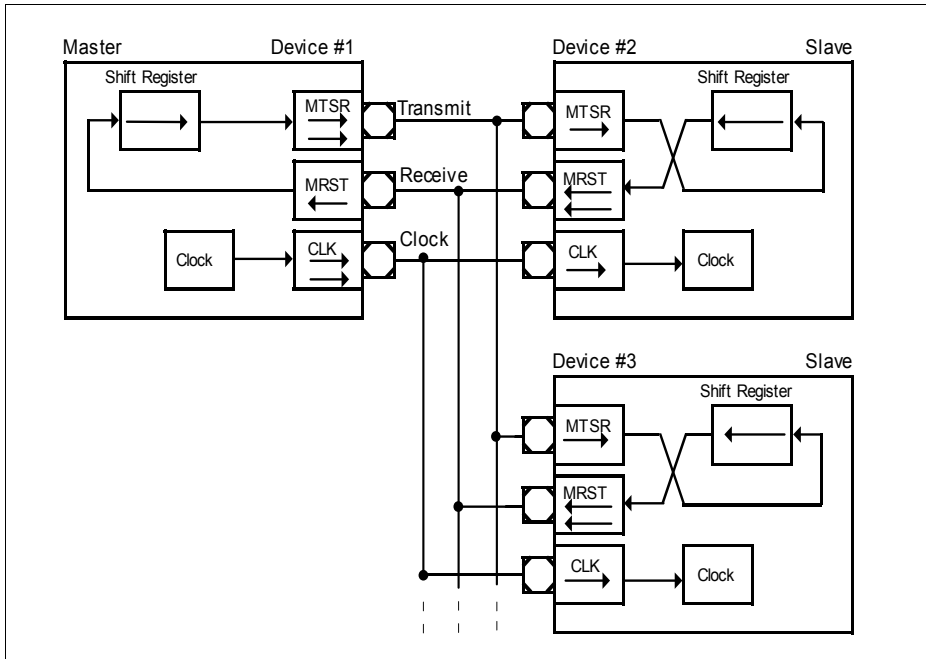


Figure 18-3 SSC Full-Duplex Configuration

The data output pins MRST of all slave devices are connected together onto the one receive line in the configuration shown in [Figure 18-3](#). During a transfer, each slave shifts out data from its shift register. There are two ways to avoid collisions on the receive line due to different slave data:

- Only one slave drives the line, i.e. enables the driver of its MRST pin. All the other slaves must have their MRST pins programmed as input so only one slave can put its data onto the master's receive line. Only receiving data from the master is possible. The master selects the slave device from which it expects data either by separate select lines, or by sending a special command to this slave. The selected slave then switches its MRST line to output until it gets a de-selection signal or command. This option is applicable only if pin has output driver disabling capability.
- The slaves use open drain output on MRST. This forms a wired-AND connection. The receive line needs an external pull-up in this case. Corruption of the data on the receive line sent by the selected slave is avoided when all slaves not selected for transmission to the master only send ones (1s). Because this high level is not actively driven onto the line, but only held through the pull-up device, the selected slave can pull this line actively to a low-level when transmitting a zero bit. The master selects

High-Speed Synchronous Serial Interface

the slave device from which it expects data either by separate select lines or by sending a special command to this slave.

After performing the necessary initialization of the SSC, the serial interfaces can be enabled. For a master device, the alternate clock line will now go to its programmed polarity. The alternate data line will go to either 0 or 1 until the first transfer starts. After a transfer, the alternate data line will always remain at the logic level of the last transmitted data bit.

When the serial interfaces are enabled, the master device can initiate the first data transfer by writing the transmit data into register TB. This value is copied into the shift register (assumed to be empty at this time), and the selected first bit of the transmit data will be placed onto the TXD line on the next clock from the baud-rate generator (transmission starts only if CON.EN = 1). Depending on the selected clock phase, a clock pulse will also be generated on the MS_CLK line. At the same time, with the opposite clock edge, the master latches and shifts in the data detected at its input line RXD. This “exchanges” the transmit data with the receive data. Because the clock line is connected to all slaves, their shift registers will be shifted synchronously with the master’s shift register — shifting out the data contained in the registers, and shifting in the data detected at the input line. After the preprogrammed number of clock pulses (via the data width selection), the data transmitted by the master is contained in all the slaves’ shift registers, while the master’s shift register holds the data of the selected slave. In the master and all slaves, the contents of the shift register are copied into the receive buffer RB and the receive interrupt line RIR is activated.

A slave device will immediately output the selected first bit (MSB or LSB of the transfer data) at line RXD when the contents of the transmit buffer are copied into the slave’s shift register. Bit CON.BSY is not set until the first clock edge at SS_CLK appears. The slave device will not wait for the next clock from the baud-rate generator, as the master does. The reason for this is that, depending on the selected clock phase, the first clock edge generated by the master may already be used to clock in the first data bit. Thus, the slave’s first data bit must already be valid at this time.

*Note: On the SSC, a transmission **and** a reception takes place at the same time, regardless of whether valid data has been transmitted or received.*

Note: The initialization of the CLK pin on the master requires some attention in order to avoid undesired clock transitions, which may disturb the other devices. Before the clock pin is switched to output mode, the clock output level will be selected in the control register CON and the alternate output be prepared via the related ALTSEL register, or the output latch must be loaded with the clock idle level.

High-Speed Synchronous Serial Interface

18.3.3 Half-Duplex Operation

In a Half-Duplex Mode, only one data line is necessary for both receiving and transmitting of data. There are two port configuration options for Half-Duplex Mode. The first option uses all the three pins but with the data exchange line connected to both the MTSR and MRST pins of each device; the shift clock line is connected to the SCLK pin. The second option uses two pins and requires the MTSR and MRST lines to be selected for the same GPIO pin, thus establishing also only a single data exchange line. The shift clock line is connected to the SCLK pin as before.

The master device controls the data transfer by generating the shift clock, while the slave devices receive it. Due to the fact that all transmit and receive pins are connected to the one data exchange line, serial data may be moved between arbitrary stations.

Similar to Full-Duplex Mode, there are two ways to avoid collisions on the data exchange line:

- Only the transmitting device may enable its transmit pin driver. This option is applicable only if pin has output driver disabling capability.
- The non-transmitting devices use open drain output and send only ones.

Because the data inputs and outputs are connected together, a transmitting device will clock in its own data at the input pin (MRST for a master device, MTSR for a slave). By this method, any corruptions on the common data exchange line are detected if the received data is not equal to the transmitted data.

High-Speed Synchronous Serial Interface

The Half-Duplex Mode port configuration using three pins is shown in [Figure 18-4](#).

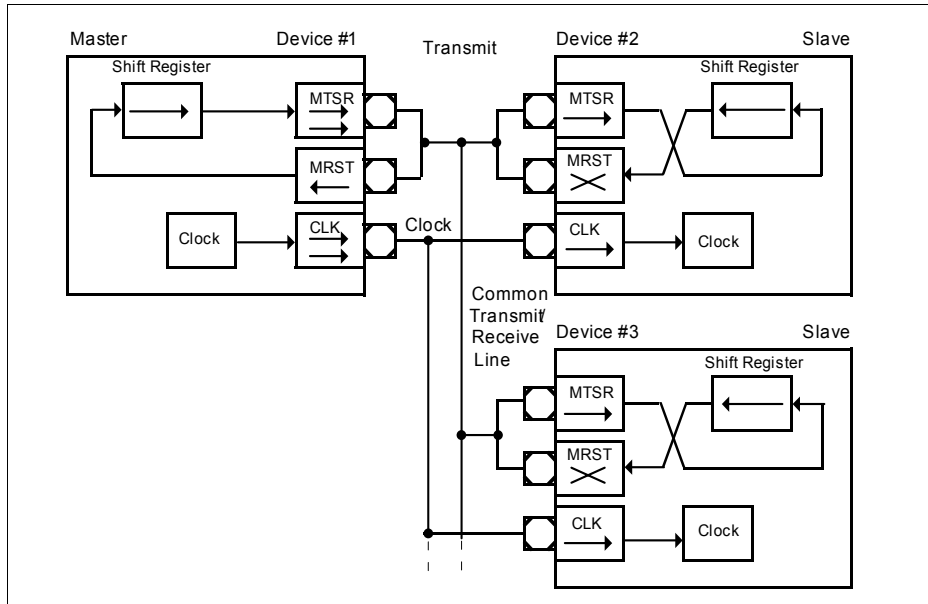


Figure 18-4 SSC Half-Duplex Configuration

18.3.4 Continuous Transfers

When the transmit interrupt request flag is set, it indicates that the transmit buffer TB is empty and ready to be loaded with the next transmit data. If TB has been reloaded by the time the current transmission is finished, the data is immediately transferred to the shift register and the next transmission will start without any additional delay. On the data line, there is no gap between the two successive frames. For example, two byte transfers would look the same as one word transfer. This feature can be used to interface with devices that can operate with or require more than 8 data bits per transfer. It is just a matter of software, how long a total data frame length can be. This option can also be used to interface to byte-wide and word-wide devices on the same serial bus, for instance.

Note: Of course, this can happen only in multiples of the selected basic data width, because it would require disabling/enabling of the SSC to reprogram the basic data width on-the-fly.

High-Speed Synchronous Serial Interface

18.3.5 Baud Rate Generation

The serial channel SSC has its own dedicated 16-bit baud-rate generator with 16-bit reload capability, allowing baud rate generation independent of the timers. **Figure 18-5** shows the baud-rate generator.

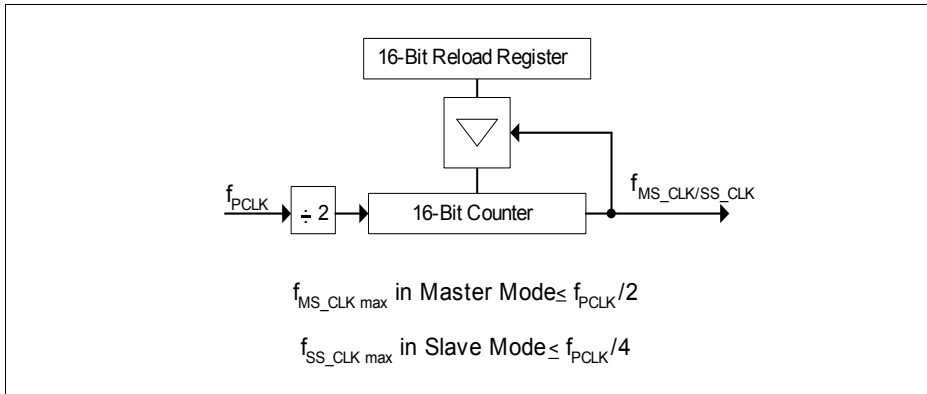


Figure 18-5 SSC Baud-rate Generator

The baud-rate generator is clocked with the module clock $f_{\text{hw_clk}}$. The timer counts downwards. Register BR is the dual-function Baud-rate Generator/Reload register. Reading BR, while the SSC is enabled, returns the contents of the timer. Reading BR, while the SSC is disabled, returns the programmed reload value. In this mode, the desired reload value can be written to BR.

Note: Never write to BR while the SSC is enabled.

The formulas below calculate either the resulting baud rate for a given reload value, or the required reload value for a given baud rate:

$$\text{Baud rate} = \frac{f_{\text{hw_clk}}}{2 \cdot (\text{BR} + 1)} \quad \text{BR} = \frac{f_{\text{hw_clk}}}{2 \cdot \text{Baud rate}} - 1 \quad (18.1)$$

 represents the contents of the reload register, taken as an unsigned 16-bit integer, while baud rate is equal to $f_{\text{MS_CLK/SS_CLK}}$ as shown in **Figure 18-5**.

The maximum baud rate that can be achieved when using a module clock of 24 MHz is 12 MBaud in Master Mode (with
 = 0000_H) or 6 MBaud in Slave Mode (with
 = 0001_H).

Table 18-3 lists some possible baud rates together with the required reload values and the resulting bit times, assuming a module clock of 24 MHz.

High-Speed Synchronous Serial Interface

Table 18-3 Typical Baud Rates of the SSC ($f_{\text{hw_clk}} = 24 \text{ MHz}$)

Reload Value	Baud Rate ($= f_{\text{MS_CLK}} / \text{SS_CLK}$)	Deviation
0000 _H	12 MBaud (only in Master Mode)	0.0%
0001 _H	6 MBaud	0.0%
0005 _H	2 MBaud	0.0%
000B _H	1 MBaud	0.0%
0017 _H	500 kBaud	0.0%
0077 _H	100 kBaud	0.0%
FFFF _H	183.11 Baud	0.0%

18.3.6 Error Detection Mechanisms

The SSC is able to detect four different error conditions. Receive Error and Phase Error are detected in all modes; Transmit Error and Baud Rate Error apply only to Slave Mode. When an error is detected, the respective error flag is/can be set and an error interrupt request will be generated by activating the EIR line (see [Figure 18-6](#)) if enabled. The error interrupt handler may then check the error flags to determine the cause of the error interrupt. The error flags are not reset automatically but rather must be cleared by software after servicing. This allows servicing of some error conditions via interrupt, while the others may be polled by software.

Note: The error interrupt handler must clear the associated (enabled) error flag(s) to prevent repeated interrupt requests.

High-Speed Synchronous Serial Interface

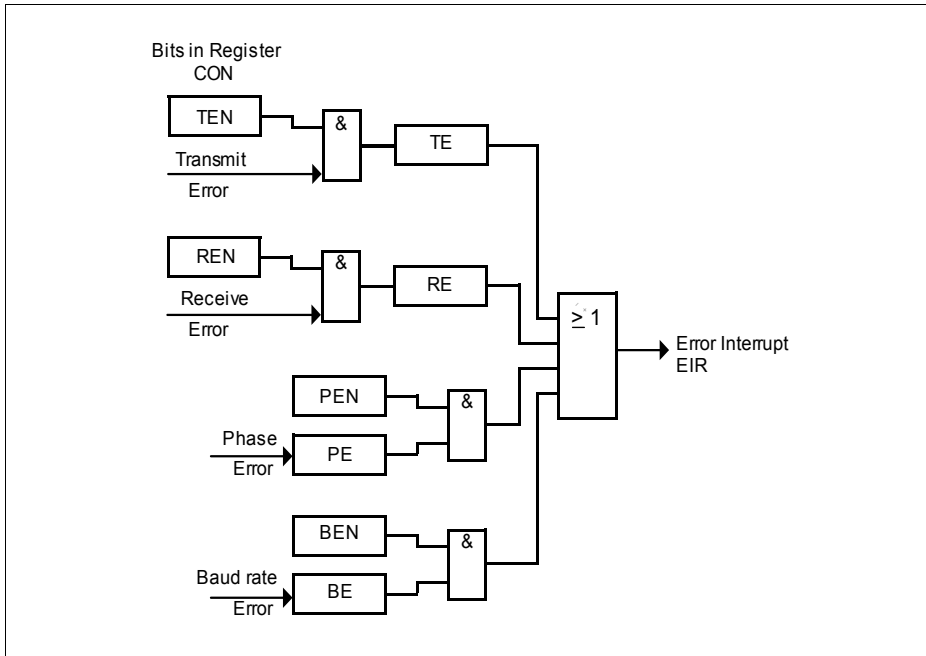


Figure 18-6 SSC Error Interrupt Control

A **Receive Error** (Master or Slave Mode) is detected when a new data frame is completely received but the previous data was not read out of the receive buffer register RB. This condition sets the error flag CON.RE and the error interrupt request line EIR, when enabled via CON.REN. The old data in the receive buffer RB will be overwritten with the new value and is irretrievably lost.

A **Phase Error** (Master or Slave Mode) is detected when the incoming data at pin MRST (Master Mode) or MTSR (Slave Mode), sampled with the same frequency as the module clock, changes between one cycle before and two cycles after the latching edge of the shift clock signal SCLK. This condition sets the error flag CON.PE and, when enabled via CON.PEN, the error interrupt request line EIR.

Note: When receiving and transmitting data in parallel, phase errors occur if the baud rate is configured to $f_{hw_clk} / 2$.

A **Baud Rate Error** (Slave Mode) is detected when the incoming clock signal deviates from the programmed baud rate by more than 100%, i.e. it is either more than double or less than half the expected baud rate. This condition sets the error flag CON.BE and, when enabled via CON.BEN, the error interrupt request line EIR. Using this error detection capability requires that the slave's baud-rate generator is programmed to the

High-Speed Synchronous Serial Interface

same baud rate as the master device. This feature detects false additional, or missing pulses on the clock line (within a certain frame).

Note: If this error condition occurs and bit CON.AREN = 1, an automatic reset of the SSC will be performed in case of this error. This is done to re-initialize the SSC if too few or too many clock pulses have been detected.

Note: This error can occur after any transfer if the communication is stopped. This is the case due to the fact that the SSC module supports back-to-back transfers for multiple transfers. In order to handle this, the baud rate detector expects after a finished transfer immediately a next clock cycle for a new transfer.

A **Transmit Error** (Slave Mode) is detected when a transfer was initiated by the master (SS_CLK gets active) but the transmit buffer TB of the slave was not updated since the last transfer. This condition sets the error flag CON.TE and the error interrupt request line EIR, when enabled via CON.TEN. If a transfer starts while the transmit buffer is not updated, the slave will shift out the 'old' contents of the shift register, which normally is the data received during the last transfer. This may lead to corruption of the data on the transmit/receive line in half-duplex mode (open drain configuration) if this slave is not selected for transmission. This mode requires that slaves not selected for transmission only shift out ones; that is, their transmit buffers must be loaded with 'FFFF_H' prior to any transfer.

Note: In order to avoid possible conflicts or misinterpretations, it is recommended to always load the slave's transmit buffer prior to any transfer.

The cause of an error interrupt request (receive, phase, baud rate, transmit error) can be identified by the error status flags in control register CON.

Note: In contrast to the error interrupt request line EIR, the error status flags CON.TE, CON.RE, CON.PE, and CON.BE, are not reset automatically upon entry into the error interrupt service routine, but must be cleared by software.

High-Speed Synchronous Serial Interface

18.4 Interrupts

The three SSC interrupts can be separately enabled or disabled by setting or clearing their corresponding enable bits in SFR SCU_MODIEN.

For a detailed description of the various interrupts see [Section 18.3](#). An overview is given in [Table 18-4](#).

Table 18-4 SSC Interrupt Sources

Interrupt	Signal	Description
Transmission starts	TIR	Indicates that the transmit buffer can be reloaded with new data.
Transmission ends	RIR	The configured number of bits have been transmitted and shifted to the receive buffer.
Receive Error	EIR	This interrupt occurs if a new data frame is completely received and the last data in the receive buffer was not read.
Phase Error	EIR	This interrupt is generated if the incoming data changes between one cycle before and two cycles after the latching edge of the shift clock signal SCLK.
Baud Rate Error (Slave Mode only)	EIR	This interrupt is generated when the incoming clock signal deviates from the programmed baud rate by more than 100%.
Transmit Error (Slave Mode only)	EIR	This interrupt is generated when TB was not updated since the last transfer if a transfer is initiated by a master.

High-Speed Synchronous Serial Interface**18.5 Register Description**

The SSC Special Function Registers are accessed from the standard (non-mapped) SFR area. The addresses of the SFRs are listed in [Table 18-5](#).

Table 18-5 Register Map

Address	Register
AA _H	CONL
AB _H	CONH
AC _H	TBL
AD _H	RBL
AE _H	BRL
AF _H	BRH

High-Speed Synchronous Serial Interface

18.5.1 Configuration Register

The operating mode of the serial channel SSC is controlled by the control register CON. This register contains control bits for mode and error check selection, and status flags for error identification. Depending on bit EN, either control functions or status flags and master/slave control are enabled.

CON.EN = 0: Programming Mode

SSC_CONL

Control Register Low [Programming Mode]

(AA_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
LB	PO	PH	HB	BM			
rw	rw	rw	rw	rw			

Field	Bit	Type	Description
BM	[3:0]	rw	Data Width Selection 0000 _B Reserved. Do not use this combination. 0001 _B - 0111 _B Transfer Data Width is 2 ... 8 bits (<BM>+1). <i>Note: The most significant bit of BM field is always fixed at 0. Thus the transfer and receive data width is restricted at maximum 8 bits. Transfer Data Width is 2...8 bits (<BM>+1).</i>
HB	4	rw	Heading Control 0 _B Transmit/Receive LSB First. 1 _B Transmit/Receive MSB First.
PH	5	rw	Clock Phase Control 0 _B Shift transmit data on the leading clock edge, latch on trailing edge. 1 _B Latch receive data on leading clock edge, shift on trailing edge.
PO	6	rw	Clock Polarity Control 0 _B Idle clock line is low, leading clock edge is low-to-high transition. 1 _B Idle clock line is high, leading clock edge is high-to-low transition.

High-Speed Synchronous Serial Interface

Field	Bit	Type	Description
LB	7	rw	Loop Back Control 0_B Normal output. 1_B Receive input is connected with transmit output (half-duplex mode).

SSC_CONH

Control Register High [Programming Mode]
(AB_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
EN	MS	0	AREN	BEN	PEN	REN	TEN
rw	rw	r	rw	rw	rw	rw	rw

Field	Bit	Type	Description
TEN	0	rw	Transmit Error Enable 0_B Ignore transmit errors. 1_B Check transmit errors.
REN	1	rw	Receive Error Enable 0_B Ignore receive errors. 1_B Check receive errors.
PEN	2	rw	Phase Error Enable 0_B Ignore phase errors. 1_B Check phase errors.
BEN	3	rw	Baud Rate Error Enable 0_B Ignore baud rate errors. 1_B Check baud rate errors.
AREN	4	rw	Automatic Reset Enable 0_B No additional action upon a baud rate error. 1_B The SSC is automatically reset upon a baud rate error.
MS	6	rw	Master Select 0_B Slave Mode. Operate on shift clock received via SCLK. 1_B Master Mode. Generate shift clock and output it via SCLK.

High-Speed Synchronous Serial Interface

Field	Bit	Type	Description
EN	7	rw	Enable Bit = 0 Transmission and reception disabled. Access to control bits.
0	5	r	Reserved Returns 0 if read; should be written with 0.

High-Speed Synchronous Serial Interface

CON.EN = 1: Operating Mode

SSC_CONL

Control Register Low [Operating Mode]

(AA_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
0				BC			
r				rh			

Field	Bit	Type	Description
BC	[3:0]	rh	Bit Count Field Shift counter is updated with every shift bit. <i>Note: This bit field is not to be written to.</i>
0	[7:4]	r	Reserved Returns 0 if read; should be written with 0.

SSC_CONH

Control Register High [Operating Mode]

(AB_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
EN	MS	0	BSY	BE	PE	RE	TE
rw	rw	r	rh	rwh	rwh	rwh	rwh

Field	Bit	Type	Description
TE	0	rwh	Transmit Error Flag 0 _B No error. 1 _B Transfer starts with the slave's transmit buffer not being updated.
RE	1	rwh	Receive Error Flag 0 _B No error. 1 _B Reception completed before the receive buffer was read.

High-Speed Synchronous Serial Interface

Field	Bit	Type	Description
PE	2	rwh	Phase Error Flag 0_B No error. 1_B Received data changes around sampling clock edge.
BE	3	rwh	Baud Rate Error Flag 0_B No error. 1_B More than factor 2 or 0.5 between slave's actual and expected baud rate.
BSY	4	rh	Busy Flag Set while a transfer is in progress. <i>Note: This bit is not to be written to.</i>
MS	6	rw	Master Select Bit 0_B Slave Mode. Operate on shift clock received via SCLK. 1_B Master Mode. Generate shift clock and output it via SCLK.
EN	7	rw	Enable Bit = 1 Transmission and reception enabled. Access to status flags and M/S control.
0	5	r	Reserved Returns 0 if read; should be written with 0.

Note: The target of an access to CON (control bits or flags) is determined by the state of CON.EN prior to the access; that is, writing C057_H to CON in programming mode (CON.EN = 0) will initialize the SSC (CON.EN was 0) and then turn it on (CON.EN = 1). When writing to CON, ensure that reserved locations receive zeros.

High-Speed Synchronous Serial Interface

18.5.2 Baud Rate Timer Reload Register

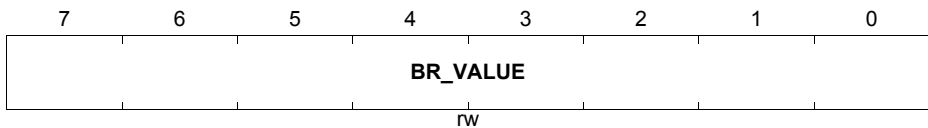
The SSC baud rate timer reload register BR contains the 16-bit reload value for the baud rate timer.

SSC_BRL

Baud Rate Timer Reload Register Low (AE_H)

Reset Value: 00_H

RMAP: 0, PAGE: X



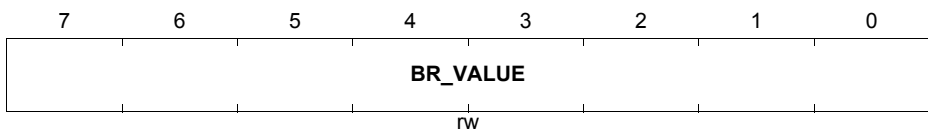
Field	Bit	Type	Description
BR_VALUE	[7:0]	rw	Baud Rate Timer/Reload Register Value Reading BR returns the 16-bit contents of the baud rate timer. Writing BR loads the baud rate timer reload register with BR_VALUE.

SSC_BRH

Baud Rate Timer Reload Register High (AF_H)

Reset Value: 00_H

RMAP: 0, PAGE: X



Field	Bit	Type	Description
BR_VALUE	[7:0]	rw	Baud Rate Timer/Reload Register Value Reading BR returns the 16-bit contents of the baud rate timer. Writing BR loads the baud rate timer reload register with BR_VALUE.

18.5.3 Transmitter Buffer Register

The SSC transmitter buffer register TB contains the transmit data value.

High-Speed Synchronous Serial Interface

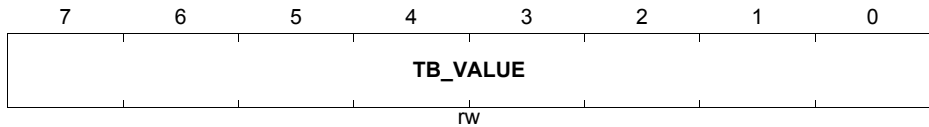
SSC_TBL

Transmitter Buffer Register Low

(AC_H)

Reset Value: 00_H

RMAP: 0, PAGE: X



Field	Bit	Type	Description
TB_VALUE	[7:0]	rw	Transmit Data Register Value TB_VALUE is the data value to be transmitted. Unselected bits of TB are ignored during transmission.

18.5.4 Receiver Buffer Register

The SSC receiver buffer register RB contains the receive data value.

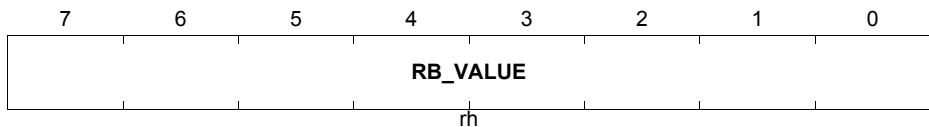
SSC_RBL

Receiver Buffer Register Low

(AD_H)

Reset Value: 00_H

RMAP: 0, PAGE: X



Field	Bit	Type	Description
RB_VALUE	[7:0]	rh	Receive Data Register Value RB contains the received data value RB_VALUE. Unselected bits of RB will be not valid and should be ignored.

19 LED and Touch-Sense Controller

This chapter describes the LEDTSCU.

19.1 Overview

The LED and Touch-sense control unit (LEDTSCU) provides for up to eight line pins and up to eight column pins, time-multiplexed control for LED driving and touchpad sensing on single pin.

LEDTSCU provides optimized HW control for column enabling and function selection. Software handling is required for line enabling per column time slice.

Features

For the LED driving function, LEDTSCU provides features:

- Selection of up to 8 LED columns; Up to 7 LED columns in case touch-sense function is enabled also
- Time slice and active duration within time slice configurable for LED column
- Possibility to drive up to 8 LEDs per time slice (column), common-cathode or common-anode
- Shadow transfer of line pattern for LED column time slice
- Interrupt for each time slice
- Line and column pins controlled by SFR setting

For the touchpad sensing function, LEDTSCU provides features:

- Up to 8 touchpad input turns
- Input pad turn controllable by software or fully by hardware round-robin
- Pad oscillation control circuit with flexible configuration, such as oscillation frequency and duration control
- 8-bit counter: For counting oscillations at pin.
- Share configuration of time slice duration with LED function, with configurable active duration within time slice
- Interrupt for each time slice, time frame
- Pin overrule control for active touch-sense function

Note: This chapter in several instances refer to the LED or touch-sense pins, e.g. 'pin COL[x]', 'pin TSIN[x]'. In all instances, it refers to the user-configured pin(s) where the alternate function (bit-field ALTSEL) selects the LED/touch-sense function. Refer to [Section 19.7](#) for more elaboration.

LED and Touch-Sense Controller
19.2 System Information

This section provides system information relevant to the LEDTSCU.

19.2.1 Pinning

Table 19-1 describes how to enable/select the particular LED or touch-sense pin function for XC82x.

Table 19-1 XC82x Pin Functions and Selection

Pin	Function	Description	Selected By
P0.0	TSIN0/LINE0	Touch-sense input 0/ LED line 0	P0_ALTSEL0.P0 = 1 _B P0_ALTSEL1.P0 = 0 _B
P0.1	TSIN1/LINE1	Touch-sense input 1/ LED line 1	P0_ALTSEL0.P1 = 1 _B P0_ALTSEL1.P1 = 0 _B
P0.2	TSIN2/LINE2	Touch-sense input 2/ LED line 2	P0_ALTSEL0.P2 = 1 _B P0_ALTSEL1.P2 = 0 _B
P0.3	TSIN3/LINE3	Touch-sense input 3/ LED line 3	P0_ALTSEL0.P3 = 1 _B P0_ALTSEL1.P3 = 0 _B
P0.4	TSIN4/LINE4	Touch-sense input 4/ LED line 4	P0_ALTSEL0.P4 = 1 _B P0_ALTSEL1.P4 = 0 _B P0_ALTSEL2.P4 = 0 _B
P0.5	TSIN5/LINE5	Touch-sense input 5/ LED line 5	P0_ALTSEL0.P5 = 1 _B P0_ALTSEL1.P5 = 0 _B P0_ALTSEL2.P5 = 0 _B
P0.6	TSIN6/LINE6	Touch-sense input 6/ LED line 6	P0_ALTSEL0.P6 = 1 _B P0_ALTSEL1.P6 = 0 _B P0_ALTSEL2.P6 = 0 _B
P1.0	COL0_0	LED column 0	P1_ALTSEL0.P0 = 1 _B P1_ALTSEL1.P0 = 0 _B
P0.4	COL0_1	LED column 0	P0_ALTSEL0.P4 = 1 _B P0_ALTSEL1.P4 = 0 _B P0_ALTSEL2.P4 = 1 _B
P1.1	COL1_0	LED column 1	P1_ALTSEL0.P1 = 1 _B P1_ALTSEL1.P1 = 0 _B
P0.5	COL1_1	LED column 1	P0_ALTSEL0.P5 = 1 _B P0_ALTSEL1.P5 = 0 _B P0_ALTSEL2.P5 = 1 _B

LED and Touch-Sense Controller
Table 19-1 XC82x Pin Functions and Selection

Pin	Function	Description	Selected By
P1.2	COL2_0	LED column 2	P1_ALTSEL0.P2 = 1 _B P1_ALTSEL1.P2 = 0 _B
P0.6	COL2_1	LED column 2	P0_ALTSEL0.P6 = 1 _B P0_ALTSEL1.P6 = 0 _B P0_ALTSEL2.P6 = 1 _B
P1.3	COL3_0	LED column 3	P1_ALTSEL0.P3 = 1 _B P1_ALTSEL1.P3 = 0 _B
P0.4	COL3_1	LED column 3	P0_ALTSEL0.P4 = 0 _B P0_ALTSEL1.P4 = 1 _B P0_ALTSEL2.P4 = 1 _B
P1.4	COL4	LED column 4	P1_ALTSEL0.P4 = 1 _B P1_ALTSEL1.P4 = 0 _B
P1.5	COL5	LED column 5	P1_ALTSEL0.P5 = 1 _B P1_ALTSEL1.P5 = 0 _B
P1.5	COLA_0	Touch-sense external pull-up / LED column A	P1_ALTSEL0.P5 = 1 _B P1_ALTSEL1.P5 = 1 _B
P0.6	COLA_1	Touch-sense external pull-up / LED column A	P0_ALTSEL0.P6 = 0 _B P0_ALTSEL1.P6 = 1 _B P0_ALTSEL2.P6 = 1 _B
P0.4	COLA_2	Touch-sense external pull-up / LED column A	P0_ALTSEL0.P4 = 1 _B P0_ALTSEL1.P4 = 1 _B P0_ALTSEL2.P4 = 1 _B

19.2.2 Clocking Configuration

There are two constant clocks provided to the LED&TSCU kernel: FPCLK (48 MHz) and SPCLK (8 MHz). The kernel is mainly running on FPCLK (48 MHz). The LEDTS-counter can be configured to run on pre-scaled clock generated from FPCLK or SPCLK. The touch-sense counter and corresponding logic is counting and processing pad oscillations on FPCLK.

If the LEDTSCU functionality is not required at all, it can be completely disabled by gating off its clock input for maximal power reduction. This is done by setting bit LTS_DIS in register PMCON1 as described below.

The bit field PAGE of SCU_PAGE register must be programmed before accessing the PMCON1 register.

LED and Touch-Sense Controller

PMCON1

Peripheral Management Control Register 1(EF_H)

Reset Value: DF_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	0	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	r	rw	rw	rw	rw	rw

Field	Bits	Type	Description
LTS_DIS	6	rw	LEDTSCU Disable Request. Active high. 0 LEDTSCU is in normal operation. 1 Request to disable the LEDTSCU. (default)
0	5	r	Reserved Returns 0 if read; should be written with 0.

19.2.3 Interrupt Events and Assignment

Table 19-2 lists the interrupt event sources from the LEDTSCU, and the corresponding event interrupt enable bit and flag bit.

Table 19-2 LEDTSCU Interrupt Events

Event	Event Interrupt Enable Bit	Event Flag Bit
Start of Time Slice	GLOBCTL1.ITS_EN	GLOBCTL1.TSF
Start of Time Frame	GLOBCTL1.ITF_EN	GLOBCTL1.TFF

Table 19-3 shows the interrupt node assignment for each LEDTSCU interrupt source.

Table 19-3 LEDTSCU Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
Start of Time Slice	IEN1.ECCIP3	–	6B _H
Start of Time Frame	IEN1.ECCIP1	–	5B _H

19.2.4 IP Interconnection

The LEDTSCU has interconnection to other peripherals enabling higher level of automation without requiring software.

Table 19-4 LEDTSCU Interconnections

LEDTSCU Function/Signal	Connected Other Module Function/Signal
Compare match (o): LEDTS_CM	ADC external trigger (i): REQTR0G, REQTR1G
Time slice interrupt (o): LEDTS_TSI	ADC external trigger (i): REQTR0H, REQTR1H

19.2.5 Debug Suspend Control

The LEDTSCU timers/counters, LEDTS-counter and TS-counter, can be enabled (together) for suspend operation when debug Monitor Mode becomes active. The debug suspend control is configurable using register MODSUSP. [Chapter 10.2.4](#) contain the detailed description of the this register.

By debug suspend, these counters stop counting (retains the last value) during the duration of the device in Monitor Mode.

19.3 Time-Multiplexed LED & Touch-Sense Functions On Pin

A single pin supports LED & touch-sense functions in time-multiplexed mode. The hardware provides the enabling for LED mode or touch-sense mode for respective function control.

In each time frame, there are maximum eight time slices configurable – up to one for touch-sense function, up to eight for LED function. Provided touch-sense function is enabled, the last time slice is reserved for touch-sense function where the pad oscillator circuit will be enabled for pin with active pad turn. Provided LED function is enabled, the rest of the time slices within the time frame are used for LED function by enabling each LED column one at a time.

As only one time slice per time frame is used for touch-sense function, depending on the number of touchpad sense inputs (pad turns) configured, by default the hardware automatically enables each pad oscillator (pad_turn_x) and sense the respective pins in round-robin fashion. Otherwise it is possible to enable for software control where the active pad turn is fully under user control. If touch-sense function is disabled, no pad turn is active in the last time slice of time frame.

A **time frame** comprises of several time slices whose duration and number is configurable. When touch-sense function is enabled for automatic hardware pad turn control, several time frames make up one **time period** where all pad_turns are completed. The time slice duration is configured centrally for the LED and/or touch-sense functions, using the LEDTS-counter. Refer to the description in [Section 19.4](#), [Section 19.8](#) and [Figure 19-2](#).

If enabled, a time slice interrupt is triggered on overflow of LEDTS-counter 8LSB for each new time slice started. A time frame interrupt may also be enabled, which is triggered on overflow of the whole LEDTS-counter.

LED and Touch-Sense Controller

To allow flexible duration of activation of LED columns and/or touch-sense oscillation counting, the duty cycle of column enable and pad oscillation enable can be adjusted for each time slice.

Figure 19-1 shows an example for a LED matrix configuration with touch pads. The configuration in this example is 8 X 4 LED matrix with 4 touch pad turns (here: 6 touch pads) enabled in sequence by hardware. Here, four time frames complete a time period.

In the time slice interrupt, software can:

- set up line value for next time slice
- set up compare value for next time slice

Refer to **Section 19.7** for **Interpretation of Bit Field FNCOL** to determine the currently active time slice.

A time frame interrupt indicates one touch input line has been sensed, application-level software can, for example:

- start touch-sense analysis routines and update status
- enable LED display update

LED and Touch-Sense Controller

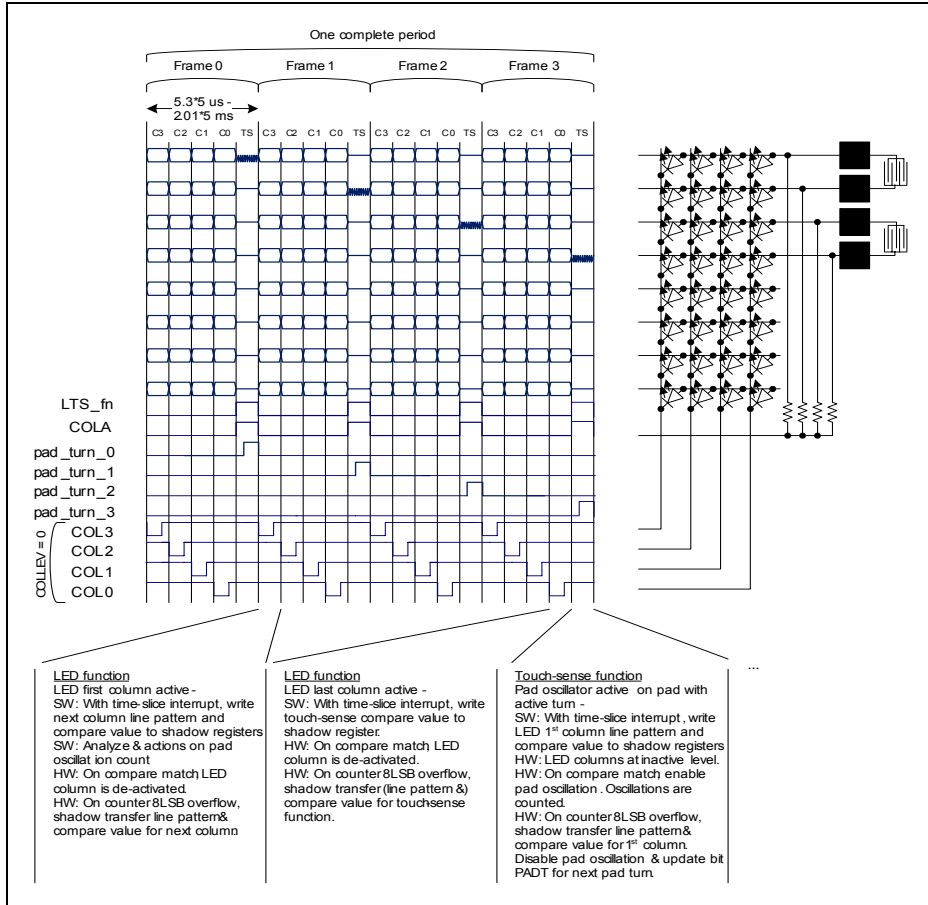


Figure 19-1 Time-Multiplexed LEDTSCU Functions on Pin (Example)

19.4 LED Driving

The control provided for LED driving is mainly for LED column selection – where one column is enabled at a time. It is required for interrupt-based software handling to control LED enabling per selected column. Up to eight LED columns are supported, and up to eight LEDs can be enabled per column.

With direct LED drive, it is supported for adjust of luminance for different types of LEDs with different forward voltages. A compare register for LEDTS-counter is provided so that the duty cycle for column enabling per time slice can be adjusted. The LED column is enabled from the beginning of the time slice until compare match event. For 100% duty cycle for LED column enable in time slice, the compare value should be set to FF_H . If the compare value is set to 00_H , the LED column will stay at passive level during the time slice. Update of the compare register for each time slice is by shadow transfer which takes place automatically at the beginning of each time slice.

A shadow transfer mechanism for update of LED enabling (LED line pattern) per column (time slice) is also provided. This shadow transfer takes place automatically at the beginning of each new time slice.

When the LEDTS-counter is first started (enable input clock by CLK_PS), a shadow transfer of line pattern and compare value is activated for the first column.

A time slice interrupt can be enabled, which occurs on overflow of the 8LSB of LEDTS-counter which indicates the start of a new time slice.

Figure 19-2 shows the LED function control circuit, which also provides the pad oscillator enable control. The clock source for the control circuit is selectable from the constant FPCLK of 48 MHz or SPCLK of 8 MHz. A 6-bit divider provides pre-scale possibilities to flexibly configure the internal LEDTS-counter count rate, which overflows in one time frame. In the duration of one time frame comprising of configurable number of time slices, the configured number of LED columns are activated in sequence. In the last time slice of the time frame, touch-sense function is activated if enabled. The time frame is divided into equal time slices of duration ranging from 5.3 μ s to 2.01 ms configurable.

The LEDTS-counter is started when bit CLK_PS is set to any other value from 0 and at least one of the LED function or the touch-sense function is/are enabled. It does not run when both functions are disabled. To avoid over-write of function enable which disturbs the hardware control during LEDTS-counter running, the TS_EN and LD_EN bits can only be modified when bit CLK_PS = 0. It is nonetheless possible to set the bits TS_EN and LD_EN in one single write to SFR GLOBCTL0 when setting CLK_PS from 0 to 1, or from 1 to 0.

When started, the counter starts running from a reset/reload value based on enabled function(s): 1) the number of columns (bit-field NR_LEDCOL) when LED function is enabled, 2) add one time slice at end of time frame when touch-sense function is enabled. The counter always counts up and overflows from 7FFH to the reload value

19.4.1 LED Pin Assignment and Current Capability

One LED column pin is enabled at a time within each configured time slice duration to control up to eight LEDs. The assignment of COL[x] to pins is configurable to suit various use cases. Refer to product data-sheet for the current capability of assigned pin/s. For example, a column pin can source/sink 20 mA nominal. This means per LED drive capability is limited to 2.5 mA when direct drive eight LEDs by column.

In all usage, the total current to direct drive (source or sink) all eight LEDs at one time must not exceed the specified maximum current for all pins (ΣI_M). For stronger LED drive capability, use of external transistors will be required.

19.5 Touchpad Sensing

Figure 19-3 shows the pin oscillation control unit, which is actually integrated with the standard GPIO pad. An active pad turn (pad_turn_x) is defined for the touch-sense input pin $\text{TSIN}[x]$ as the duration within the touch-sense time slice where the TS-counter is counting.

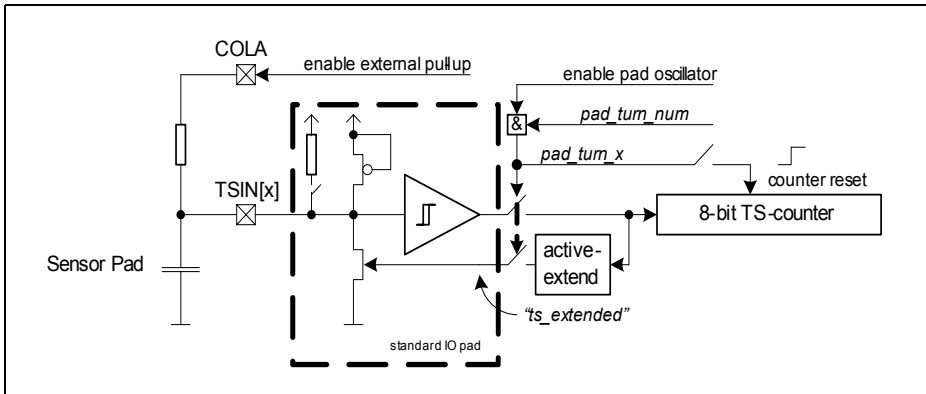


Figure 19-3 Touch-Sense Oscillator Control Circuit

The 8-bit TS-counter counts the number of oscillations. It can only be written when there is no active pad turn. The counter may be enabled for automatic reset (to 00_H) on each start of a new pad turn. Bit TSCTROVF indicates that the counter has overflowed. Alternatively, it can be configured such that the TS-counter stops counting in current time slice when the count value saturates, i.e. does not overflow & stops at FF_H . In this case, the TS-counter starts running again only on a new pad turn.

A configurable pin-low-level active extension is provided for adjustment of oscillation per user system. The extension is active during the discharge phase of oscillation, and is configurable to extend by one or four FPCLK . **Figure 19-4** illustrates this function.

The pad configuration of the active touch-sense pin $\text{TSIN}[x]$ is over-ruled by hardware in the active duration to enable oscillations, reference **Section 19.9**. In particular, the weak internal pull-up can be optionally disabled (GPIO pin SFR setting for pull applies instead) such as when the user system utilize external resistor for pull-up instead. In the whole duration of the touch-sense time slice, COLA is activated high. This activates a pull-up via an external resistor connected to pin COLA. This configuration provides some flexibility to adjust the pad oscillation rate for adaptation to user system.

The touch-sense function is time-multiplexed with the LED function on enabled $\text{LINE}[x]/\text{TSIN}[x]$ pins. During the touch-sense time slice for the other TSIN pins which are not on active pad turn, the corresponding LINE output remains active. Software should take care to set the line bits to 1 to avoid current sink from pin COLA.

LED and Touch-Sense Controller

The touch-sense function is active in the last time slice of a time frame. Refer to [Section 19.3](#), and [Section 19.4](#) for more details on time slice allocation and configuration.

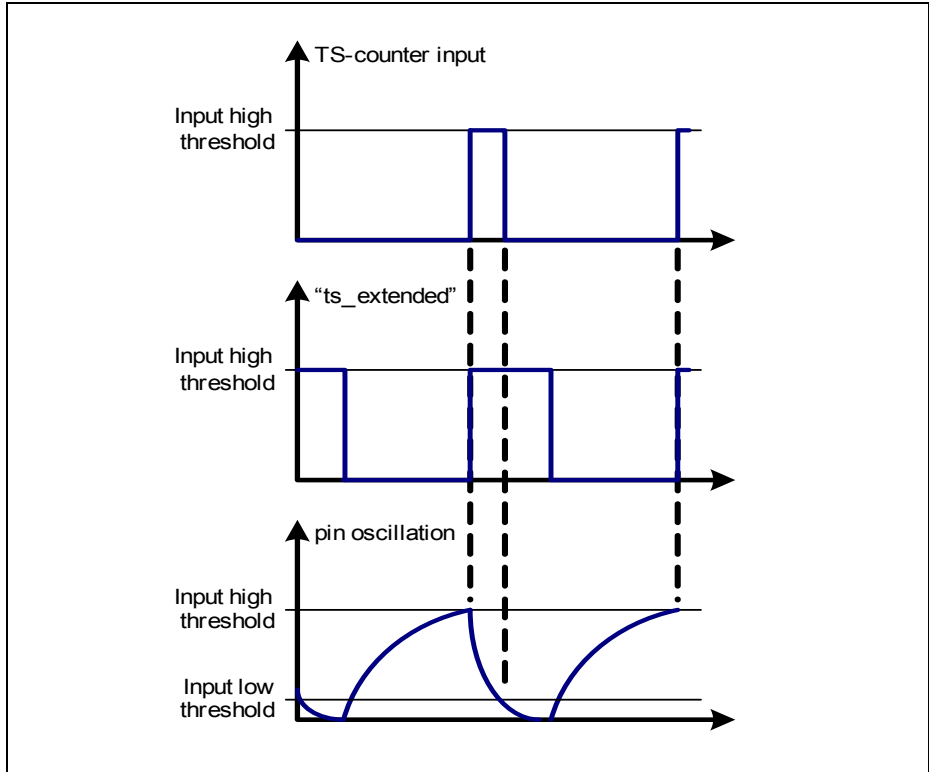


Figure 19-4 Pin-Low-Level Extension Function

The pad oscillation is enabled on pad with valid turn for configurable duration. A compare value provides to adjust the duty cycle within the time slice when the pad oscillation is enabled (TS-counter is counting). The pad oscillation is enabled only on compare match till the end of the time slice. For 100% duty cycle pad oscillation enable in time slice, the compare value shall be set to 00_H . Setting the compare value to FF_H results in no pad oscillation in time slice.

The time slice interrupt and/or time frame interrupt may be enabled as required for touch-sense control.

19.5.1 Finger Sensing

When a finger is placed on the sensor pad, it increases the pad capacitance and frequency of oscillation on pad is reduced. The pad oscillation frequency without finger is typically expected in the approximate range 0.25 MHz to 0.5 MHz. With finger on the sensor, the pad oscillation frequency is typically expected in the approximate range 0.125 MHz to 0.25 MHz. As described in above section, some flexibility is provided to adjust the oscillation in the user system: 1) Configurable pin low-level active extension, 2) Alternative enabling of external pull-up with resistance selectable by user. With a time slice duration configurable ranging from 5.3 μ s (0.18 MHz) to 2.01 ms (496 Hz), the software can configure the duration of the pad turn (adjustable within time slice using compare function) and set a count threshold for oscillations to detect if finger control is available.

19.6 Time-Multiplexed LED and Touch-Sense Function on Pin

Some hints are provided regarding the time-multiplexed usage of a pin for LED and touch-sense function:

- The maximum number of LED columns = 7 when touch-sense function is also enabled.
- If enabled by pin, COLA outputs HIGH to enable external R (resistor) as pull-up for touch-sense function.
- During touch-sense time slice, recommend to set LED lines to output HIGH.
- During LED time slice, COLA outputs LOW and will sink current if connected lines output HIGH.
- The effective capacitance for each TSIN line depends largely on what is connected to the line and the application board layout. All touch-pads for the application should be calibrated for robust touch-detection.

LED and Touch-Sense Controller

19.7 Function Enabling and Control Hints

It is recommended to set up all configuration for the LEDTSCU in all SFRs before write **LTS_GLOBCTL0** SFR to enable and start LED and/or touch-sense function(s).

Note: SFR bits especially affecting the LEDTS-counter configuration for LED/touch-sense function can only be written when the counter is not running i.e. CLK_PS = 0. Refer to SFR bit description [Section 19.11](#).

Enable LED Function Only

To enable LED function only: set LD_EN, clear TS_EN.

Initialization after reset:

```
MOV LTS_GLOBCTL0, #0b10XXXXXX
    ;set LD_EN and start LEDTS-counter on prescaled clock
    ;(CLK_PS != 0)
```

Re-configuration during run-time:

```
MOV LTS_GLOBCTL0, #0x00;stop LEDTS-counter by clearing prescaler
MOV LTS_GLOBCTL0, #0b10XXXXXX
```

Enable Touch-Sense Function Only

To enable touch-sense function only: clear LD_EN, set TS_EN.

Initialization after reset:

```
MOV LTS_GLOBCTL0, #0b01XXXXXX
    ;set TS_EN and start LEDTS-counter on prescaled clock
    ;(CLK_PS != 0)
```

Re-configuration during run-time:

```
MOV LTS_GLOBCTL0, #0x00;stop LEDTS-counter by clearing prescaler
MOV LTS_GLOBCTL0, #0b01XXXXXX
```

Enable Both LED and Touch-Sense Function

To enable both functions: set LD_EN, set TS_EN.

Initialization after reset:

```
MOV LTS_GLOBCTL0, #0b11XXXXXX
    ;set TS_EN and start LEDTS-counter on prescaled clock
    ;(CLK_PS != 0)
```

Re-configuration during run-time:

```
MOV LTS_GLOBCTL0, #0x00;stop LEDTS-counter by clearing prescaler
MOV LTS_GLOBCTL0, #0b01XXXXXX
```

LED and Touch-Sense Controller

Interpretation of Bit Field FNCOL

The handling by software in each time slice includes to update the line value and compare value to be activated (shadow-transferred) in the next time slice. The FNCOL bit field provides information on the function/column active in the previous time slice. With this information, software can determine the active function/column in current time slice and prepare the necessary values (to be shadow-transferred) valid for the next time slice.

Following is an example where six time slices are enabled (per time frame), with five LED columns and touch-sensing enabled:

Table 19-5 Interpretation of FNCOL Bit Field

FNCOL	Active Function / Column in Current Time Slice	SW Prepare via Shadow Registers for Function / Column of Next Time Slice
111 _H	LED COL[4]	LED COL[3]
010 _H	LED COL[3]	LED COL[2]
011 _H	LED COL[2]	LED COL[1]
100 _H	LED COL[1]	LED COL[0]
101 _H	LED COL[0]	Touch-sense TSIN[PADT]
110 _H	Touch-sense TSIN[PADT]	LED COL[4]

19.8 LEDTSCU Timing Calculations

LEDTSCU main timing or duration formulation are provided in following.

Count-Rate (CR):

$$CR = (fCLK) \div (PREscaler) \quad (19.1)$$

Time slice duration (TSD):

$$TSD = 2^8 \div (CR) \quad (19.2)$$

Time frame duration:

$$TimeFrameDuration = (\text{Number of time slice}) \times TSD \quad (19.3)$$

LED and Touch-Sense Controller

LED drive active duration:

(19.4)

$$\text{LED Drive Active Duration} = \text{TSD} \times \text{Compare_VALUE} \div 2^8$$

Touch-sense drive active duration:

(19.5)

$$\text{Touch-sense Drive Active Duration} = \text{TSD} \times (2^8 - \text{Compare_VALUE}) \div 2^8$$

LED and Touch-Sense Controller

19.9 LEDTSCU Pin Control

The user may flexibly assign pins as provided by GPIO-SFR ALTSEL, for the LEDTSCU functions:

- COL[x] (for LED column control)
- LINE[x]/TSIN[x] (for LED line control or touch-sensing)

Refer also to [Section 19.4](#) for more considerations with regards to which COL[x] and/or LINE[x]/TSIN[x] will be active based on user configuration.

User code must configure the ALTSEL-assigned LED pin GPIO-SFR setting for the LED function. For the touch-sense function, it is also required to configure the ALTSEL to select the TSIN (and COLA) function even as LEDTSCU provides some pin over-rule controls to the assigned touch-sense pin with active pad turn, see [Table 19-6](#) and [Figure 19-5](#).

Table 19-6 LEDTSCU Pin Control Signals

Function	LD/TS_en	LTS_fn	Pin	Control of Assigned Pin
LED column	LD_EN = 1	0 = LED	Enable COL[x]; Passive level on COL[the rest]. If TS_EN = 1, COLA = 0.	GPIO SFR setting
LED line	LD_EN = 1	0 = LED	LINE[x] = shadow-transferred from LDLINE	GPIO SFR setting
Touch-sense	TS_EN = 1	1 = Touch-sense	Enable TSIN[x] for oscillation. All other TSIN pins output LINE value. Passive level on COL[the rest] except COLA = 1.	Hardware over-rule on pad_turn_x ¹⁾ for active duration: - Enable pull-up (this over-rule can be disabled by bit EPULL) - Enable open-drain

1) For the other pad inputs not on turn, there is no HW over-rule which means the GPIO SFR setting is active.

LED and Touch-Sense Controller

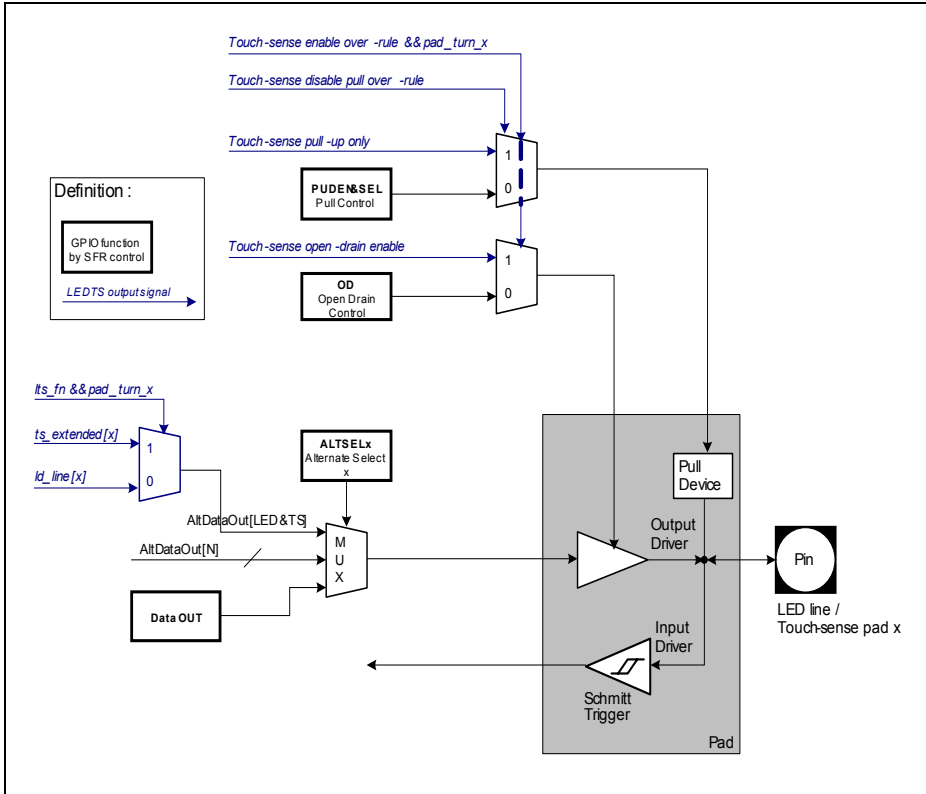


Figure 19-5 Over-rule Control on Pad-Input Pin for Touch-Sense Function

19.10 Interrupt

There are two interrupts triggered by LEDTSCU kernel: 1) time slice event, 2) time frame event. The flags are set on event or when CLK_PS is set from 0 regardless of whether the corresponding interrupt is enabled or not. When enabled, the event (including setting of CLK_PS from 0) activates the corresponding interrupt request from the kernel.

19.11 Registers Description

The **LEDTSCU** Special Function Registers are accessed from the standard (non-mapped) SFR area. [Table 19-7](#) lists the SFRs and corresponding address.

Table 19-7 Register Map

Address	Register
97 _H	LTS_GLOBCTL0
D4 _H	LTS_COMPARE
D5 _H	LTS_LDLINE
D6 _H	LTS_LDTCTL
D7 _H	LTS_TSCTL
D8 _H	LTS_GLOBCTL1
D9 _H	LTS_TSVAL

LED and Touch-Sense Controller

19.11.1 Global Control and Status

There are three registers for global control and status.

LTS_GLOBCTL0

Global Control Register 0

(97_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
LD_EN	TS_EN	CLK_PS					
rw	rw	rw					

Field	Bits	Type	Description
CLK_PS	[5:0]	rw	LEDTS-Counter Clock Pre-Scale Factor The input clock FPCLK or SPCLK is prescaled according to setting. 0 _D No clock 1 _D Divide by 1 n _D Divide by n 63 _D Divide by 63 This bit can only be set to any other value (from 0) provided at least one of touch-sense or LED function is enabled. The LEDTS-counter starts running on the input clock from reset/reload value based on enabled function(s) (and NR_LEDCOL). Refer Section 19.4 for details. When this bit is clear to 0 from other value, the LEDTS-counter stops running and resets.
TS_EN ¹⁾	6	rw	Touch-Sense Function Enable Set to enable the kernel for touch-sense function control when CLK_PS is set from 0.
LD_EN ¹⁾	7	rw	LED Function Enable Set to enable the kernel for LED function control when CLK_PS is set from 0.

1) This bit can only be modified when bit CLK_PS = 0.

LED and Touch-Sense Controller

LTS_GLOBCTL1

Global Control Register 1

(D8_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
TSF	ITS_EN	TFF	ITF_EN	CLKSEL	FNCOL		
rwh	rw	rwh	rw	rw	rh		

Field	Bits	Type	Description
FNCOL	[2:0]	rh	Previous Active Function/LED Column Status Shows the active function / LED column in the previous time slice. Updated on start of new time-slice when LEDTS-counter 8LSB over-flows. Controlled by latched value of the internal DE-MUX, see Figure 19-2 .
CLKSEL ¹⁾	3	rw	LEDTS-Counter Clock Input Select 0 _B 48 MHz is selected 1 _B 8 MHz is selected
ITF_EN	4	rw	Enable Time-Frame Interrupt 0 _B Disabled 1 _B Enabled
TFF	5	rwh	Time-Frame Interrupt Flag Set on activation of each new time-frame, including when bit CLK_PS is set from 0. To be cleared by software.
ITS_EN	6	rw	Enable Time Slice Interrupt 0 _B Disabled 1 _B Enabled
TSF	7	rwh	Time Slice Interrupt Flag Set on activation of each new time slice, including when bit CLK_PS is set from 0. To be cleared by software.

1) This bit can only be modified when bit CLK_PS = 0.

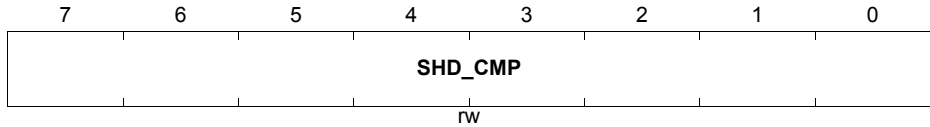
LED and Touch-Sense Controller

LTS_COMPARE

Time Slice Compare Shadow Register(D4_H)

Reset Value: 00_H

RMAP: 0, PAGE: X



Field	Bits	Type	Description
SHD_CMP	[7:0]	rw	Compare Value for Time Slice to Shadow-Transfer This value is shadow-transferred to the time slice compare register on start of each new time slice. A shadow transfer is effected on CLK_PS set from 0.

LED and Touch-Sense Controller

19.11.2 Function Control Registers

These registers provide controls for the LED drive and touch-sense functions.

LTS_LDTCTL

LED and Touch-Sense Control Register(D6_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
NR_LEDCOL			COLLEV	NR_PADT			TSOEXT
rw			rw	rw			rw

Field	Bits	Type	Description
TSOEXT	0	rw	Extension for Touch-Sense Output for Pin-Low-Level 0 _B Extend by 1 FPCLK 1 _B Extend by 4 FPCLK
NR_PADT ¹⁾	[3:1]	rw	Number of Touch-Sense Pad Turns Defines the number of touch-sense inputs, which is equal to number of pad turns. Used for the hardware control of pad turn enabling. 0 _D 1 n _D n+1
COLLEV	4	rw	Active Level of LED Column 0 _B Active low 1 _B Active high

LED and Touch-Sense Controller

Field	Bits	Type	Description
NR_LEDCOL ¹⁾	[7:5]	rw	Number of LED Columns Defines the number of LED columns. 000 _B 1 LED column 001 _B 2 LED columns 010 _B 3 LED columns 011 _B 4 LED columns 100 _B 5 LED columns 101 _B 6 LED columns 110 _B 7 LED columns 111 _B 8 LED columns (max. LED columns = 7 if bit TS_EN = 1) <i>Note: LED column is enabled in sequence starting from highest column number. If touch-sense function is not enabled, COLA is activated in last time slice.</i>

1) This bit can only be modified when bit CLK_PS = 0.

LTS_LDLINE

LED Line Pattern Shadow Register (D5_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
SHD_LINE							
rw							

Field	Bits	Type	Description
SHD_LINE	[7:0]	rw	LED Line Value to Shadow-Transfer This value is shadow-transferred to the LED lines on start of each new time slice. A shadow transfer is effected on CLK_PS set from 0.

LED and Touch-Sense Controller

LTS_TSCTL

Touch-Sense Control Register

(D7_H)

Reset Value: 00_H

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
TSCTROV F	TSCTRR	TSCTRSA T	EPULL	PADTSW	PADT		
rwh	rw	rw	rw	rw	rwh		

Field	Bits	Type	Description
PADT	[2:0]	rwh	Touch-Sense Pad Turn (Input Pin) This is the next or currently active pad turn (touch-sense input pin). When PADTSW = 0, the value is updated by hardware at the end of touch-sense time slice. Software write is always possible. 0 _D TSIN0 n _D TSIN[n]
PADTSW¹⁾	3	rw	Software Control for Touch-Sense Pad Turn 0 _B The hardware automatically enables the touch-sense inputs round-robin, starting from TSIN0. 1 _B Disable hardware control for software control only. The active touch-sense input is configured in bit PADT.
EPULL	4	rw	Enable External Pull-up Configuration on Pin COLA When set, the internal pull-up over-rule on active touch-sense input pin is disabled. 0 _B HW over-rule to enable internal pull-up is active on TSIN[x] for set duration in touch-sense time slice. With this setting, it is not specified to assign the COLA to any pin. 1 _B Enable external pull-up: Output 1on pin COLA for whole duration of touch-sense time slice. <i>Note: Independent of this setting, COLA always outputs 1 for whole duration of touch-sense time slice.</i>

LED and Touch-Sense Controller

Field	Bits	Type	Description
TSCTRSAT	5	rw	Saturation of TS-Counter 0_B Disable 1_B Enable. TS-counter stops counting in the current time slice when it reaches FF_H . Counter starts to count again on new pad turn, triggered on compare match.
TSCTRR	6	rw	TS-Counter Auto Reset 0_B Disable TS-counter automatic reset 1_B Enable TS-counter automatic reset to 00_H on new pad turn. Triggered on compare match in time slice.
TSCTROVF	7	rwh	TS-Counter Overflow Indication This bit indicates whether a TS-counter overflow has occurred. This bit is cleared on new pad turn, triggered on compare match. 0_B No overflow has occurred. 1_B The TS-counter has overflowed at least once.

1) This bit can only be modified when bit CLK_PS = 0.

LTS_TSVAL
Touch-Sense Counter Value
(D9_H)
Reset Value: 00_H
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
TSCTRVAL							
rwh							

Field	Bits	Type	Description
TSCTRVAL	[7:0]	rwh	TS-Counter Value This shows the actual TS-counter value. It can only be written when no pad turn is active. This counter may be enabled for automatic reset on the start of a new pad turn.

20 Capture/Compare Unit 6 (CCU6)

The CCU6 is a high-resolution 16-bit capture and compare unit with application specific modes, mainly for AC drive control. Special operating modes support the control of Brushless DC-motors using Hall sensors or Back-EMF detection. Furthermore, block commutation and control mechanisms for multi-phase machines are supported.

It also supports inputs to start several timers synchronously, an important feature in devices with several CCU6 modules.

This chapter is structured as follows:

- Introduction (see [Section 20.1](#))
including the register overview (see [Section 20.1.3](#))
- System information (see [Section 20.2](#))
- Operating T12 (see [Section 20.3](#))
including T12 related registers (see [Section 20.3.8](#))
and capture/compare control registers (see [Section 20.3.9](#))
- Operating T13 (see [Section 20.4](#))
including T13 related registers (see [Section 20.4.6](#))
- Trap handling (see [Section 20.5](#))
- Multi-Channel mode (see [Section 20.6](#))
- Hall sensor mode (see [Section 20.7](#))
- Modulation control registers (see [Section 20.8](#))
- Interrupt handling (see [Section 20.9](#))
including interrupt registers (see [Section 20.9.2](#))
- General module operation (see [Section 20.10](#))
including general registers (see [Section 20.10.2](#))

20.1 Introduction

The CCU6 unit is made up of a Timer T12 Block with three capture/compare channels and a Timer T13 Block with one compare channel. The T12 channels can independently generate PWM signals or accept capture triggers, or they can jointly generate control signal patterns to drive AC-motors or inverters.

A rich set of status bits, synchronized updating of parameter values via shadow registers, and flexible generation of interrupt request signals provide means for efficient software-control.

*Note: The capture/compare module itself is named CCU6 (capture/compare unit 6).
A capture/compare channel inside this module is named CC6x.*

20.1.1 Feature Set Overview

This section gives an overview over the different building blocks and their main features.

Timer 12 Block Features

- Three capture/compare channels, each channel can be used either as capture or as compare channel
- Generation of a three-phase PWM supported (six outputs, individual signals for high-side and low-side switches)
- 16-bit resolution, maximum count frequency = peripheral clock
- Dead-time control for each channel to avoid short-circuits in the power stage
- Concurrent update of T12 registers
- Center-aligned and edge-aligned PWM can be generated
- Single-shot mode supported
- Start can be controlled by external events
- Capability of counting external events
- Many interrupt request sources
- Hysteresis-like control mode

Timer 13 Block Features

- One independent compare channel with one output
- 16-bit resolution, maximum count frequency = peripheral clock
- Concurrent update of T13 registers
- Can be synchronized to T12
- Interrupt generation at period-match and compare-match
- Single-shot mode supported
- Start can be controlled by external events
- Capability of counting external events

Additional Specific Functions

- Block commutation for Brushless DC-drives implemented
- Position detection via Hall-sensor pattern
- Noise filter supported for position input signals
- Automatic rotational speed measurement and commutation control for block commutation
- Integrated error handling
- Fast emergency stop without CPU load via external signal ($\overline{\text{CTRAP}}$)
- Control modes for multi-channel AC-drives
- Output levels can be selected and adapted to the power stage

Capture/Compare Unit 6 (CCU6)

20.1.2 Block Diagram

The Timer T12 can operate in capture and/or compare mode for its three channels. The modes can also be combined (e.g. a channel operates in compare mode, whereas another channel operates in capture mode). The Timer T13 can operate in compare mode only. The multi-channel control unit generates output patterns which can be modulated by T12 and/or T13. The modulation sources can be selected and combined for the signal modulation.

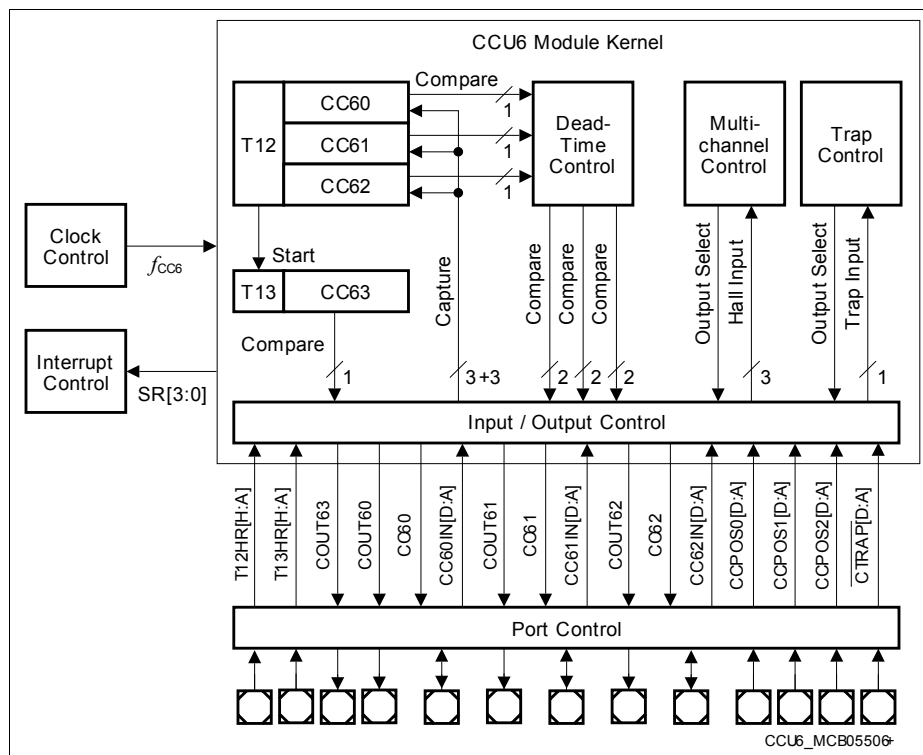


Figure 20-1 CCU6 Block Diagram

Capture/Compare Unit 6 (CCU6)

20.1.3 Register Overview

For the generation of the overall register table, the prefix “CCU6x_” has to be added to the register names in this table to identify the registers of different CCU6 modules that are implemented. In this naming convention, x indicates the module number.

Table 20-1 shows all registers required for programming of a CCU6 module. It summarizes the CCU6 kernel registers and defines the offset and the reset values. 8-bit short addresses are not available for this module.

T12 related Registers	Cap/Com Control Registers	Interrupt Status/ Control Registers	General Registers
T12L	CMPSTATL	ISL	PISEL0L
T12H	CMPSTATH	ISH	PISEL0H
T12PRL	CMPMODIFL	ISSL	PISEL2
T12PRH	CMPMODIFH	ISSH	
T12DTCL	T12MSELL	ISRL	
T12DTCH	T12MSELH	ISRH	
CC60RL	TCTR0L	INPL	
CC60RH	TCTR0H	INPH	
CC60SRL	TCTR2L	IENL	
CC60SRH	TCTR2H	IENH	
CC61RL	TCTR4L		
CC61RH	TCTR4H		
CC61SRL	Modulation Control Registers		
CC61SRH	MODCTRL		
CC62RL	MODCTRH		
CC62RH	TRPCTRL		
CC62SRL	TRPCTRH		
CC62SRH	PSLR		
T13 related Registers		MCMCTR	
T13L		MCMOUTSL	
T13H		MCMOUTSH	
T13PRL		MCMOUTL	
T13PRH		MCMOUTH	
CC63RL			
CC63RH			
CC63SRL			
CC63SRH			

8-Bit_CCU6_regs

Figure 20-2 CCU6 Registers

Capture/Compare Unit 6 (CCU6)

Table 20-1 CCU6 Module Register Summary

Short Name	Description	Offset	Reset Value	See Page
General Registers				
PISEL0L	Port Input Select Register Low	9E _H	00 _H	Page 20-127
PISEL0H	Port Input Select Register High	9F _H	00 _H	Page 20-128
PISEL2	Port Input Select Register 2	A4 _H	00 _H	Page 20-129

Timer T12 related Registers

T12L	Timer 12 Counter Register Low	FA _H	00 _H	Page 20-43
T12H	Timer 12 Counter Register High	FB _H	00 _H	Page 20-43
T12PRL	Timer 12 Period Register Low	9C _H	00 _H	Page 20-44
T12PRH	Timer 12 Period Register High	9D _H	00 _H	Page 20-44
T12DTCL	Dead-Time Control Register for Timer T12 Low	A4 _H	00 _H	Page 20-49
T12DTCH	Dead-Time Control Register for Timer T12 High	A5 _H	00 _H	Page 20-49
CC60RL	Capture/Compare Register Channel CC60 Low	FA _H	00 _H	Page 20-46
CC60RH	Capture/Compare Register Channel CC60 High	FB _H	00 _H	Page 20-46
CC61RL	Capture/Compare Register Channel CC61 Low	FC _H	00 _H	Page 20-46
CC61RH	Capture/Compare Register Channel CC61 High	FD _H	00 _H	Page 20-46
CC62RL	Capture/Compare Register Channel CC62 Low	FE _H	00 _H	Page 20-46
CC62RH	Capture/Compare Register Channel CC62 High	FF _H	00 _H	Page 20-46
CC60SRL	Capture/Compare Shadow Register Channel CC60 Low	FA _H	00 _H	Page 20-47
CC60SRH	Capture/Compare Shadow Register Channel CC60 High	FB _H	00 _H	Page 20-47

Capture/Compare Unit 6 (CCU6)
Table 20-1 CCU6 Module Register Summary (cont'd)

Short Name	Description	Offset	Reset Value	See Page
CC61SRL	Capture/Compare Shadow Register Channel CC61 Low	FC _H	00 _H	Page 20-47
CC61SRH	Capture/Compare Shadow Register Channel CC61 High	FD _H	00 _H	Page 20-47
CC62SRL	Capture/Compare Shadow Register Channel CC62 Low	FE _H	00 _H	Page 20-47
CC62SRH	Capture/Compare Shadow Register Channel CC62 High	FF _H	00 _H	Page 20-47

Capture/Compare Control Registers

CMPSTATL	Compare State Register Low	FE _H	00 _H	Page 20-51
CMPSTATH	Compare State Register High	FF _H	00 _H	Page 20-52
CMPMODIF L	Compare State Modification Register Low	A6 _H	00 _H	Page 20-53
CMPMODIF H	Compare State Modification Register High	A7 _H	00 _H	Page 20-54
T12MSELL	T12 Capture/Compare Mode Select Register Low	9A _H	00 _H	Page 20-55
T12MSELH	T12 Capture/Compare Mode Select Register High	9B _H	00 _H	Page 20-55
TCTR0L	Timer Control Register 0 Low	A6 _H	00 _H	Page 20-57
TCTR0H	Timer Control Register 0 High	A7 _H	00 _H	Page 20-59
TCTR2L	Timer Control Register 2 Low	FA _H	00 _H	Page 20-61
TCTR2H	Timer Control Register 2 High	FB _H	00 _H	Page 20-63
TCTR4L	Timer Control Register 4 Low	9C _H	00 _H	Page 20-64
TCTR4H	Timer Control Register 4 High	9D _H	00 _H	Page 20-65

Timer T13 related Registers

T13L	Timer 13 Counter Register Low	FC _H	00 _H	Page 20-79
T13H	Timer 13 Counter Register High	FD _H	00 _H	Page 20-79
T13PRL	Timer 13 Period Register Low	9E _H	00 _H	Page 20-81
T13PRH	Timer 13 Period Register High	9F _H	00 _H	Page 20-81

Capture/Compare Unit 6 (CCU6)

Table 20-1 CCU6 Module Register Summary (cont'd)

Short Name	Description	Offset	Reset Value	See Page
CC63RL	Compare Register for Timer 13 Low	9A _H	00 _H	Page 20-83
CC63RH	Compare Register for Timer 13 High	9B _H	00 _H	Page 20-83
CC63SRL	Compare Shadow Register for Timer 13 Low	9A _H	00 _H	Page 20-84
CC63SRH	Compare Shadow Register for Timer 13 Low	9B _H	00 _H	Page 20-84

Modulation Control Registers

MODCTRL	Modulation Control Register Low	FC _H	00 _H	Page 20-98
MODCTRH	Modulation Control Register High	FD _H	00 _H	Page 20-99
TRPCTRL	Trap Control Register Low	FE _H	00 _H	Page 20-100
TRPCTRH	Trap Control Register High	FF _H	00 _H	Page 20-101
PSLR	Passive State Level Register	A6 _H	00 _H	Page 20-103
MCMOUTS L	Multi-Channel Mode Output Shadow Register Low	9E _H	00 _H	Page 20-106
MCMOUTS H	Multi-Channel Mode Output Shadow Register High	9F _H	00 _H	Page 20-106
MCMOUTL	Multi-Channel Mode Output Register Low	9A _H	00 _H	Page 20-107
MCMOUTH	Multi-Channel Mode Output Register High	9B _H	00 _H	Page 20-109
MCMCTR	Multi-Channel Mode Control Register	A7 _H	00 _H	Page 20-104

Interrupt Status and Node Registers

ISL	Interrupt Status Register Low	9C _H	00 _H	Page 20-112
ISH	Interrupt Status Register High	9D _H	00 _H	Page 20-113
ISSL	Interrupt Status Set Register Low	A4 _H	00 _H	Page 20-116
ISSH	Interrupt Status Set Register High	A5 _H	00 _H	Page 20-116
ISRL	Interrupt Status Reset Register Low	A4 _H	00 _H	Page 20-118
ISRH	Interrupt Status Reset Register High	A5 _H	00 _H	Page 20-118
INPL	Interrupt Node Pointer Register Low	9E _H	40 _H	Page 20-124
INPH	Interrupt Node Pointer Register High	9F _H	39 _H	Page 20-125

Capture/Compare Unit 6 (CCU6)

Table 20-1 CCU6 Module Register Summary (cont'd)

Short Name	Description	Offset	Reset Value	See Page
IENL	Interrupt Node Pointer Register Low	9C _H	0000 _H	Page 20-120
IENH	Interrupt Node Pointer Register High	9D _H	0000 _H	Page 20-121

Note: In the case of a write access to addresses inside the address range (that is covered by the same chip select signal), but that are not the addresses explicitly mentioned for the module, the write access is not taken into account for the module. The same principle is valid for read accesses. In case of a read access to another address, the module does not react.

20.2 System Information

This section provides system information relevant to the CCU6.

20.2.1 Pinning

The CCU6 pin assignment for XC82x is shown in [Table 20-2](#).

Table 20-2 CCU6 Pin Functions and Selection

Pin	Function	Description	Selected By
Capture Input Signals			
P1.1	CC60_0	Input signals for capture event on channel CC60	CCU6_PISEL0L.ISCC60 = 00 _B
P0.3	CC60_1		CCU6_PISEL0L.ISCC60 = 01 _B
P1.3	CC61_0	Input signals for capture event on channel CC61	CCU6_PISEL0L.ISCC61 = 00 _B
P0.1	CC61_1		CCU6_PISEL0L.ISCC61 = 01 _B
P1.5	CC62_0	Input signals for capture event on channel CC62	CCU6_PISEL0L.ISCC62 = 00 _B
P0.2	CC62_1		CCU6_PISEL0L.ISCC62 = 01 _B
Trap Input Signals			
P0.3	CTRAP_0	Input signals for CTRAP	MODPISEL3.CTRAPIS = 00 _B CCU6_PISEL0L.ISTRP = 00 _B
P0.4	CTRAP_1		MODPISEL3.CTRAPIS = 01 _B CCU6_PISEL0L.ISTRP = 00 _B
P2.3	CTRAP_2		MODPISEL3.CTRAPIS = 10 _B CCU6_PISEL0L.ISTRP = 00 _B
Hall Input Signals			

Capture/Compare Unit 6 (CCU6)
Table 20-2 CCU6 Pin Functions and Selection

Pin	Function	Description	Selected By
P0.0	CCPOS0_0	Input signals for CCPOS0	CCU6_PISEL0H.ISPOS0 = 00 _B
P2.0	CCPOS0_1		CCU6_PISEL0H.ISPOS0 = 01 _B
P2.3	CCPOS0_2		CCU6_PISEL0H.ISPOS0 = 10 _B
P0.1	CCPOS1_0	Input signals for CCPOS1	CCU6_PISEL0H.ISPOS1 = 00 _B
P2.1	CCPOS1_1		CCU6_PISEL0H.ISPOS1 = 01 _B
P0.2	CCPOS2_0	Input signals for CCPOS2	CCU6_PISEL0H.ISPOS2 = 00 _B
P2.2	CCPOS2_1		CCU6_PISEL0H.ISPOS2 = 01 _B

Timer Input Signals

P0.0	T12HR_0	Input signals for T12HR	MODPISEL3.IST12HR1 = 000 _B CCU6_PISEL0H.IST12HR = 00 _B
P2.0	T12HR_2		MODPISEL3.IST12HR1 = 010 _B CCU6_PISEL0H.IST12HR = 00 _B
P2.2	T12HR_3		MODPISEL3.IST12HR1 = 011 _B CCU6_PISEL0H.IST12HR = 00 _B
P0.1	T13HR_0	Input signals for T13HR	MODPISEL3.IST13HR1 = 000 _B CCU6_PISEL2.IST13HR = 00 _B
P0.0	T13HR_1		MODPISEL3.IST13HR1 = 000 _B CCU6_PISEL2.IST13HR = 00 _B
P2.0	T13HR_2		MODPISEL3.IST13HR1 = 000 _B CCU6_PISEL2.IST13HR = 00 _B
P2.2	T13HR_3		MODPISEL3.IST13HR1 = 000 _B CCU6_PISEL2.IST13HR = 00 _B

Compare Output Signals

P1.1	CC60_0	Compare outputs for channel CC60	P1_ALTSEL0.P1 = 0 _B P1_ALTSEL1.P1 = 1 _B
P0.3	CC60_1		P0_ALTSEL0.P3 = 1 _B P0_ALTSEL1.P3 = 1 _B
P1.0	COU60_0		P1_ALTSEL0.P0 = 0 _B P1_ALTSEL1.P0 = 1 _B

Capture/Compare Unit 6 (CCU6)

Table 20-2 CCU6 Pin Functions and Selection

Pin	Function	Description	Selected By
P1.3	CC61_0	Compare outputs for channel CC61	P1_ALTSEL0.P3 = 0 _B P1_ALTSEL1.P3 = 1 _B
P0.1	CC61_1		P0_ALTSEL0.P1 = 1 _B P0_ALTSEL1.P1 = 1 _B
P1.2	COUT61_0		P1_ALTSEL0.P2 = 0 _B P1_ALTSEL1.P2 = 1 _B
P0.0	COUT61_1		P0_ALTSEL0.P0 = 1 _B P0_ALTSEL1.P0 = 1 _B
P1.5	CC62_0	Compare outputs for channel CC62	P1_ALTSEL0.P5 = 0 _B P1_ALTSEL1.P5 = 1 _B
P0.2	CC62_1		P0_ALTSEL0.P2 = 1 _B P0_ALTSEL1.P2 = 1 _B
P1.4	COUT62_0		P1_ALTSEL0.P4 = 0 _B P1_ALTSEL1.P4 = 1 _B
P0.5	COUT62_1		P0_ALTSEL0.P5 = 1 _B P0_ALTSEL1.P5 = 1 _B P0_ALTSEL2.P5 = 0 _B
P1.2	COUT63_0	Compare outputs for channel CC63	P1_ALTSEL0.P2 = 1 _B P1_ALTSEL1.P2 = 1 _B
P1.4	COUT63_1		P1_ALTSEL0.P4 = 1 _B P1_ALTSEL1.P4 = 1 _B

The bit field PAGE of SCU_PAGE register must be programmed before accessing the MODPSEL3 register.

MODPSEL3

Peripheral Input Select Register 3 (EE_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
IST13HR1			IST12HR1			CTRAPIS	
rw			rw			rw	

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
CTRAPIS	[1:0]	rw	CCU6 CTRAP Input Selection 00 CCU6 CTRAP Input Pin CTRAP_0 is selected. 01 CCU6 CTRAP Input Pin CTRAP_1 is selected. 10 CCU6 CTRAP Input Pin CTRAP_2 is selected. 11 Out of Range Channel 3 event on Input Pin CTRAP_3 is selected. <i>Note: To select the port pin/ORC event to trigger CTRAP, CCU6_PISEL0L.ISTRP must be set to 00_B.</i>
IST12HR1	[4:2]	rw	CCU6 T12HR Port Pin Input Select 000 T12HR Input Pin T12HR_0 is selected. 001 Out of Range Channel 0 event on Input T12HR_1 is selected. 010 T12HR Input Pin T12HR_2 is selected. 011 T12HR Input Pin T12HR_3 is selected. 100 Out of Range Channel 2 event on Input T12HR_4 is selected. 101 Reserved. 110 Reserved. 111 Reserved. <i>Note: To select the port pin/ORC event to trigger T12HR, CCU6_PISEL0H.IST12HR must be set to 00_B.</i>

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
IST13HR1	[7:5]	rw	CCU6 T13HR Port Pin Input Select 000 T13HR Input Pin T13HR_0 is selected. 001 T13HR Input Pin T13HR_1 is selected. 010 T13HR Input Pin T13HR_2 is selected. 011 T13HR Input Pin T13HR_3 is selected. 100 Out of Range Channel 2 event on Input T13HR_4 is selected. 101 Reserved. 110 Reserved. 111 Reserved. <i>Note: To select the port pin/ORC event to trigger T13HR, CCU6_PISEL2.IST13HR must be set to 00_B.</i>

20.2.2 Clocking Configuration

The CCU6 kernel runs on the FPCLK at a fixed frequency of 48 MHz.

If the CCU6 functionality is not required at all, it can be completely disabled by gating off its clock input for maximal power reduction. This is done by setting bit CCU_DIS in register PMCON1 as described below.

The bit field PAGE of SCU_PAGE register must be programmed before accessing the PMCON1 register.

PMCON1

Peripheral Management Control Register 1(EF_H)

Reset Value: DF_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	0	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	r	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CCU_DIS	2	rw	CCU6 Disable Request. Active high. 0 CCU6 is in normal operation. 1 Request to disable the CCU6. (default)
0	5	r	Reserved Returns 0 if read; should be written with 0.

Capture/Compare Unit 6 (CCU6)

20.2.3 Interrupt Events and Assignment

Table 20-3 lists the interrupt event sources from the CCU6, and the corresponding event interrupt enable bit and flag bit.

Table 20-3 CCU6 Interrupt Events

Event	Event Interrupt Enable Bit	Event Flag Bit	Service Request Output
See Section 20.9	See IENL , IENH	See ISL , ISH	SR0, SR1, SR2, SR3

Table 20-4 shows the interrupt node assignment for each CCU6 interrupt source.

Table 20-4 CCU6 Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
SR0	IEN1.ECCIP0	IRCON2.CCU6SR0	53 _H
SR1	IEN1.ECCIP1	IRCON2.CCU6SR1	5B _H
SR2	IEN1.ECCIP2 MODIEN.CCU6SR2EN	IRCON3.CCU6SR2	63 _H
SR3	IEN1.ECCIP3 MODIEN.CCU6SR3EN	IRCON3.CCU6SR3	6B _H

Beside the event interrupt enable bit as shown in [Table 20-3](#), bit MODIEN.CCU6SR2EN and MODIEN.CCU6SR3EN must be set to 1_B to enable the SR2 and SR3 interrupt service request. The bit field PAGE of register SCU_PAGE must be programmed before accessing the MODIEN register.

Note: SR2 and SR3 event of CCU6 module are used as interconnection output signals to trigger CC6x inputs, T12HR input and T13HR input. If no interrupt service is required during this type of trigger mode, control bits in register MODIEN can be disabled. However, for SR2 and SR3 events to be used as interconnection output signal, the event interrupt enable bit in register IENL and IENH must be enabled.

Capture/Compare Unit 6 (CCU6)

MODIEN

Peripheral Interrupt Enable Register (F7_H)

Reset Value: 07_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
CCU6SR3 EN	CCU6SR2 EN	0			RIREN	TIREN	EIREN
rw	rw	r			rw	rw	rw

Field	Bits	Type	Description
CCU6SR2EN	6	rw	CCU6 SR2 Enable 0 CCU6 SR2 interrupt service request is disabled 1 CCU6 SR2 interrupt service request is enabled
CCU6SR3EN	7	rw	CCU6 SR3 Enable 0 CCU6 SR3 interrupt service request is disabled 1 CCU6 SR3 interrupt service request is enabled
0	[5:3]	r	Reserved Returns 0 if read; should be written with 0.

Capture/Compare Unit 6 (CCU6)
20.2.4 IP Interconnection

The CCU6 has interconnection to other peripherals enabling higher level of automation without requiring software.

Table 20-5 CCU6 Outputs Interconnection

CCU6 Function/Signal	Connected Other Module Function/Signal
Timer 12 output signals	
T12 period match (o): T12PM	ADC Request Source 0/1 Trigger 2 Inputs (i): REQTR0C, REQTR1C
Timer 13 output signals	
T13 compare match (o): T13CM	ADC Request Source 0/1 Trigger 4 Inputs (i): REQTR0E, REQTR1E
T13 period match (o): T13PM	ADC Request Source 0/1 Trigger 3 Inputs (i): REQTR0D, REQTR1D
CCU6 service request output signals	
Service request output SR2 (o): CCU6_SR2	ADC Request Source 0/1 Trigger 0 Input (i): REQTR0A, REQTR1A
Service request output SR3 (o): CCU6_SR3	ADC Request Source 0/1 Trigger 1 Input (i): REQTR0B, REQTR1B
Miscellaneous signals	
MCM shadow transfer (o): MCM_ST	ADC Request Source 0/1 Trigger 5 Inputs (i): REQTR0F, REQTR1F

Table 20-6 CCU6 Inputs Interconnection

CCU6 Function/Signal	Connected Other Module Function/Signal	Selected By
Capture Inputs		
CCU6 input (i): CC60	CCU6 SR2 output (o): CCU6_SR2	CCU6_PISEL0L.ISCC60 = 10 _B
	ADC boundary event 0 (o): ADC_BF0	CCU6_PISEL0L.ISCC60 = 11 _B
CCU6 input (i): CC61	CCU6 SR2 output (o): CCU6_SR2	CCU6_PISEL0L.ISCC61 = 10 _B
	ADC boundary event 1 (o): ADC_BF1	CCU6_PISEL0L.ISCC61 = 11 _B

Capture/Compare Unit 6 (CCU6)
Table 20-6 CCU6 Inputs Interconnection

CCU6 Function/Signal	Connected Other Module Function/Signal	Selected By
CCU6 input (i): CC62	CCU6 SR2 output (o): CCU6_SR2	CCU6_PISEL0L.ISCC62 = 10 _B
	ADC Boundary event 2 (o): ADC_BF2	CCU6_PISEL0L.ISCC62 = 11 _B
CCU6 input (i): T12HR	CCU6 SR2 output (o): CCU6_SR2	CCU6_PISEL0H.IST12HR = 01 _B
	CCU6 SR3 output (o): CCU6_SR3	CCU6_PISEL0H.IST12HR = 10 _B
	ADC Channel event (o): ADC_CHEV	CCU6_PISEL0H.IST12HR = 11 _B
	ORC event 0 (o): ORCEVENT0	MODPISEL3.IST12HR1 = 001 _B CCU6_PISEL0H.IST12HR = 00 _B
	ORC event 2 (o): ORCEVENT2	MODPISEL3.IST12HR1 = 100 _B CCU6_PISEL0H.IST12HR = 00 _B
CCU6 input (i): T13HR	CCU6 SR2 output (o): CCU6_SR2	CCU6_PISEL2.IST13HR = 01 _B
	CCU6 SR3 output (o): CCU6_SR3	CCU6_PISEL2.IST13HR = 10 _B
	ADC Channel event (o): ADC_CHEV	CCU6_PISEL2.IST13HR = 11 _B
	ORC event 2 (o): ORCEVENT2	MODPISEL3.IST13HR1 = 100 _B CCU6_PISEL2.IST13HR = 00 _B
CCU6 input (i): CTRAP	ORC event 3 (o): ORCEVENT3	MODPISEL3.CTRAPIS = 11 _B CCU6_PISEL0L.ISTRP = 00 _B
	ADC Channel event 0 (o): ADC_CHEV0	CCU6_PISEL0L.ISTRP = 11 _B
CCU6 input (i): CCPOS0	ADC boundary event 0 (o): ADC_BF0	CCU6_PISEL0H.ISPOS0 = 11 _B
CCU6 input (i): CCPOS1	ADC boundary event 1 (o): ADC_BF1	CCU6_PISEL0H.ISPOS1 = 11 _B
CCU6 input (i): CCPOS2	ADC boundary event 2 (o): ADC_BF2	CCU6_PISEL0H.ISPOS2 = 11 _B

20.2.5 Module Suspend Control

When the On-Chip Debug Support (OCDS) is in Monitor Mode (MMCR2.MMODE = 1) and the Debug-Suspend signal is active (MMCR2.DSUSP = 1), Timer T12 and Timer T13 of CCU6 module in XC82x can be suspended based on the settings of their corresponding module suspend bits in register MODSUSP. The definition of this register is described in [Chapter 10.2.4](#).

When suspended, only the timer stops counting as the counter input clock is gated off. The module is still clocked so that module registers are accessible.

20.3 Operating Timer T12

The timer T12 block is the main unit to generate the 3-phase PWM signals. A 16-bit counter is connected to 3 channel registers via comparators, that generate a signal when the counter contents match one of the channel register contents. A variety of control functions facilitate the adaptation of the T12 structure to different application needs. Besides the 3-phase PWM generation, the T12 block offers options for individual compare and capture functions, as well as dead-time control and hysteresis-like compare mode.

This section provides information about:

- T12 overview (see [Section 20.3.1](#))
- Counting scheme (see [Section 20.3.2](#))
- Compare modes (see [Section 20.3.3](#))
- Compare mode output path (see [Section 20.3.4](#))
- Capture modes (see [Section 20.3.5](#))
- Shadow transfer (see [Section 20.3.6](#))
- T12 operating mode selection (see [Section 20.3.7](#))
- T12 counter register description (see [Section 20.3.8](#))

Capture/Compare Unit 6 (CCU6)

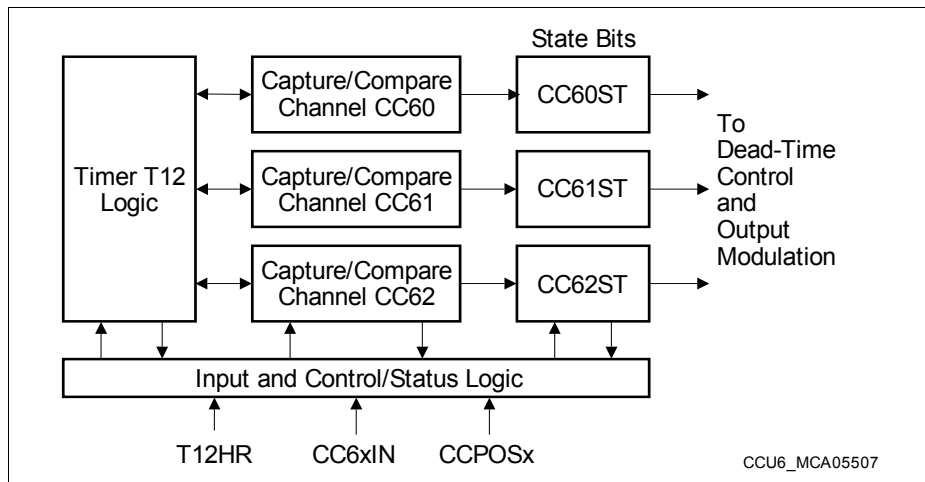


Figure 20-3 Overview Diagram of the Timer T12 Block

20.3.1 T12 Overview

Figure 20-4 shows a detailed block diagram of Timer T12. The functions of the timer T12 block are controlled by bits in registers **TCTR0L**, **TCTR0H**, **TCTR2L**, **TCTR2H**, **TCTR4L**, **TCTR4H**, **PISEL0L** and **PISEL0H**.

Timer T12 receives its input clock (f_{T12}) from the module clock f_{CC6} via a programmable prescaler and an optional 1/256 divider or from an input signal T12HR. These options are controlled via bit fields T12CLK and T12PRE (see **Table 20-7**). T12 can count up or down, depending on the selected operation mode. A direction flag, CDIR, indicates the current counting direction.

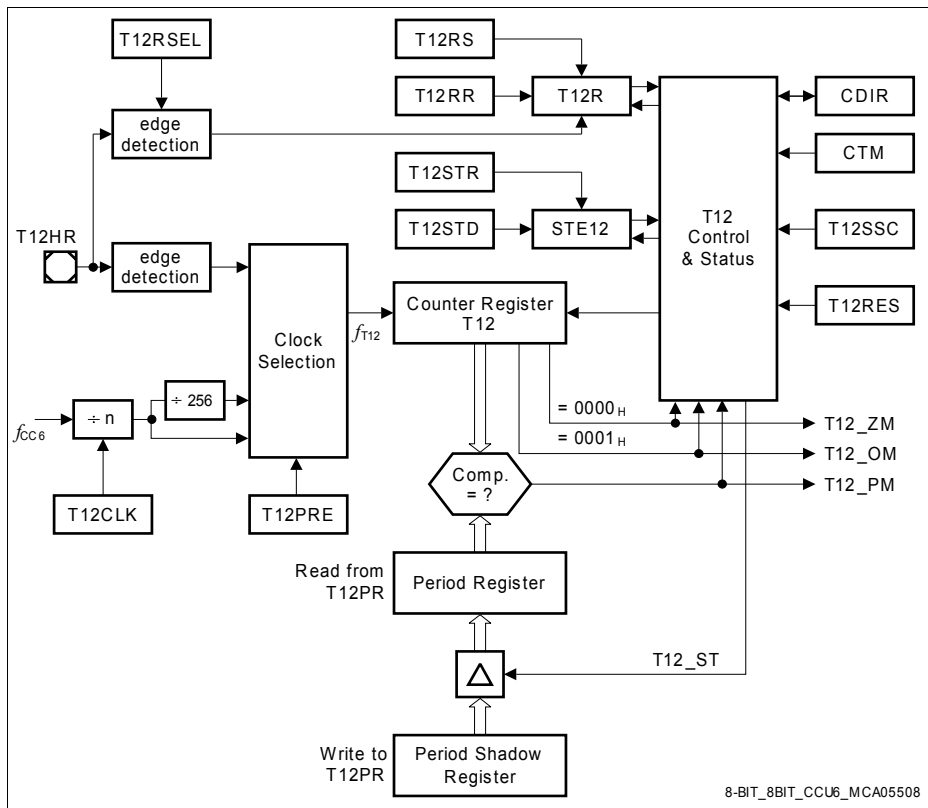


Figure 20-4 Timer T12 Logic and Period Comparators

Via a comparator, the T12 counter register **T12L**, **T12H** is connected to a Period Register **T12PRL**, **T12PRH**. This register determines the maximum count value for T12. In Edge-Aligned mode, T12 is cleared to 0000_H after it has reached the period value

Capture/Compare Unit 6 (CCU6)

defined by T12PR. In Center-Aligned mode, the count direction of T12 is set from 'up' to 'down' after it has reached the period value (please note that in this mode, T12 exceeds the period value by one before counting down). In both cases, signal T12_PM (T12 Period Match) is generated. The Period Register receives a new period value from its Shadow Period Register.

A read access to T12PR delivers the current period value at the comparator, whereas a write access targets the Shadow Period Register to prepare another period value. The transfer of a new period value from the Shadow Period Register into the Period Register (see [Section 20.3.6](#)) is controlled via the 'T12 Shadow Transfer' control signal, T12_ST. The generation of this signal depends on the operating mode and on the shadow transfer enable bit STE12. Providing a shadow register for the period value as well as for other values related to the generation of the PWM signal allows a concurrent update by software for all relevant parameters.

Two further signals indicate whether the counter contents are equal to 0000_H (T12_ZM = zero match) or 0001_H (T12_OM = one match). These signals control the counting and switching behavior of T12.

The basic operating mode of T12, either Edge-Aligned mode ([Figure 20-5](#)) or Center-Aligned mode ([Figure 20-6](#)), is selected via bit CTM. A Single-Shot control bit, T12SSC, enables an automatic stop of the timer when the current counting period is finished (see [Figure 20-7](#) and [Figure 20-8](#)).

The start or stop of T12 is controlled by the Run bit T12R that can be modified by bits in register [TCTR4L](#), [TCTR4H](#). The run bit can be set/cleared by software via the associated set/clear bits T12RS or T12RR, it can be set by a selectable edge of the input signal T12HR ([TCTR2H](#).T12RSEL), or it is cleared by hardware according to preselected conditions.

The timer T12 run bit T12R must not be set while the applied T12 period value is zero. Timer T12 can be cleared via control bit T12RES. Setting this write-only bit does only clear the timer contents, but has no further effects, for example, it does not stop the timer.

The generation of the T12 shadow transfer control signal, T12_ST, is enabled via bit STE12. This bit can be set or reset by software indirectly through its associated set/clear control bits T12STR and T12STD.

While Timer T12 is running, write accesses to the count register T12 are not taken into account. If T12 is stopped and the Dead-Time counters are 0, write actions to register T12 are immediately taken into account.

20.3.2 T12 Counting Scheme

This section describes the clocking and counting capabilities of T12.

20.3.2.1 Clock Selection

The input clock f_{T12} of Timer T12 is derived from the internal module clock f_{CC6} through a programmable prescaler and an optional 1/256 divider. The resulting prescaler factors are listed in [Table 20-7](#). The prescaler of T12 is cleared while T12 is not running (**TCTRL.T12R** = 0) to ensure reproducible timings and delays.

Table 20-7 Timer T12 Input Frequency Options

T12CLK	Resulting Input Clock f_{T12} Prescaler Off (T12PRE = 0)	Resulting Input Clock f_{T12} Prescaler On (T12PRE = 1)
000 _B	f_{CC6}	$f_{CC6} / 256$
001 _B	$f_{CC6} / 2$	$f_{CC6} / 512$
010 _B	$f_{CC6} / 4$	$f_{CC6} / 1024$
011 _B	$f_{CC6} / 8$	$f_{CC6} / 2048$
100 _B	$f_{CC6} / 16$	$f_{CC6} / 4096$
101 _B	$f_{CC6} / 32$	$f_{CC6} / 8192$
110 _B	$f_{CC6} / 64$	$f_{CC6} / 16384$
111 _B	$f_{CC6} / 128$	$f_{CC6} / 32768$

20.3.2.2 Edge-Aligned / Center-Aligned Mode

In **Edge-Aligned Mode** (CTM = 0), timer T12 is always counting upwards (CDIR = 0). When reaching the value given by the period register (period-match T12_PM), the value of T12 is cleared with the next counting step (saw tooth shape).

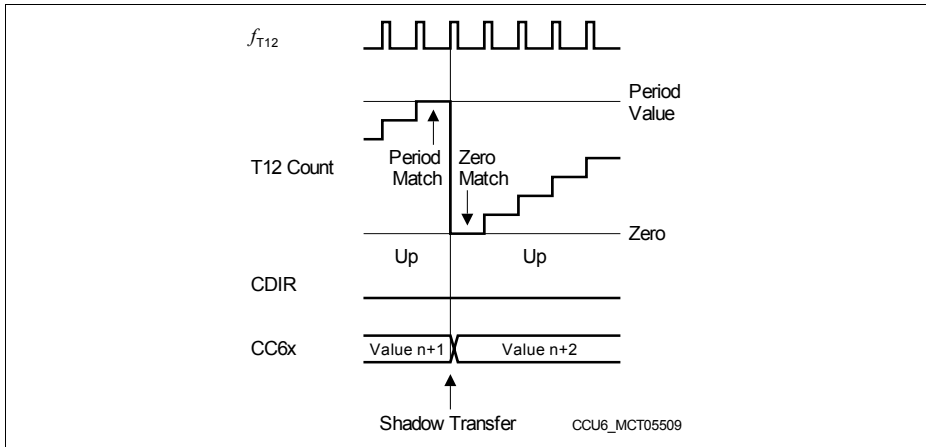


Figure 20-5 T12 Operation in Edge-Aligned Mode

As a result, in Edge-Aligned mode, the timer period is given by:

$$T12_{PER} = \text{<Period-Value>} + 1; \text{ in } T12 \text{ clocks } (f_{T12}) \quad (20.1)$$

In **Center-Aligned Mode** (CTM = 1), timer T12 is counting upwards or downwards (triangular shape). When reaching the value given by the period register (period-match T12_PM) while counting upwards (CDIR = 0), the counting direction control bit CDIR is changed to downwards (CDIR = 1) with the next counting step.

When reaching the value 0001_H (one-match T12_OM) while counting downwards, the counting direction control bit CDIR is changed to upwards with the next counting step.

As a result, in Center.Aligned mode, the timer period is given by:

$$T12_{PER} = (\text{<Period-Value>} + 1) \times 2; \text{ in } T12 \text{ clocks } (f_{T12}) \quad (20.2)$$

- With the next clock event of f_{T12} the count direction is set to counting up (CDIR = 0) when the counter reaches 0001_H while counting down.
- With the next clock event of f_{T12} the count direction is set to counting down (CDIR = 1) when the Period-Match is detected while counting up.
- With the next clock event of f_{T12} the counter counts up while CDIR = 0 and it counts down while CDIR = 1.

Capture/Compare Unit 6 (CCU6)

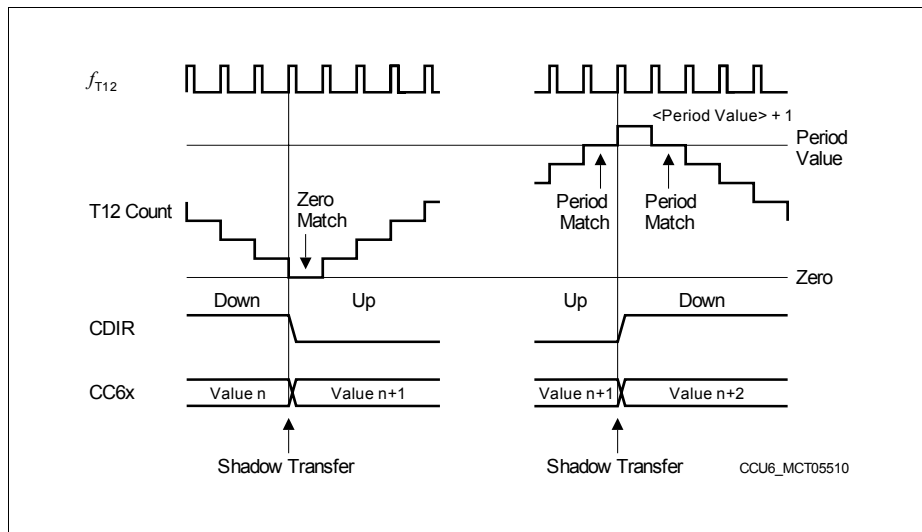


Figure 20-6 T12 Operation in Center-Aligned Mode

Note: Bit CDIR changes with the next timer clock event after the one-match or the period-match. Therefore, the timer continues counting in the previous direction for one cycle before actually changing its direction (see [Figure 20-6](#)).

20.3.2.3 Single-Shot Mode

In Single-Shot Mode, the timer run bit T12R is cleared by hardware. If bit T12SSC = 1, the timer T12 will stop when the current timer period is finished.

In Edge-Aligned mode, T12R is cleared when the timer becomes zero after having reached the period value (see [Figure 20-7](#)).

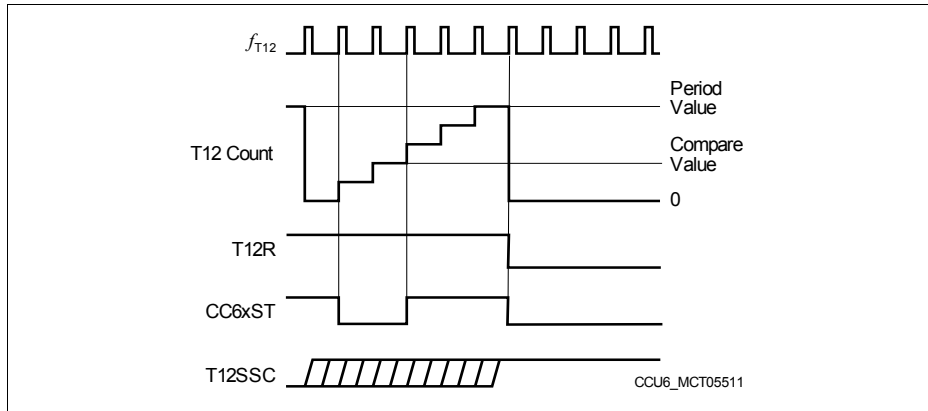


Figure 20-7 Single-Shot Operation in Edge-Aligned Mode

In Center-Aligned mode, the period is finished when the timer has counted down to zero (one clock cycle after the one-match while counting down, see [Figure 20-8](#)).

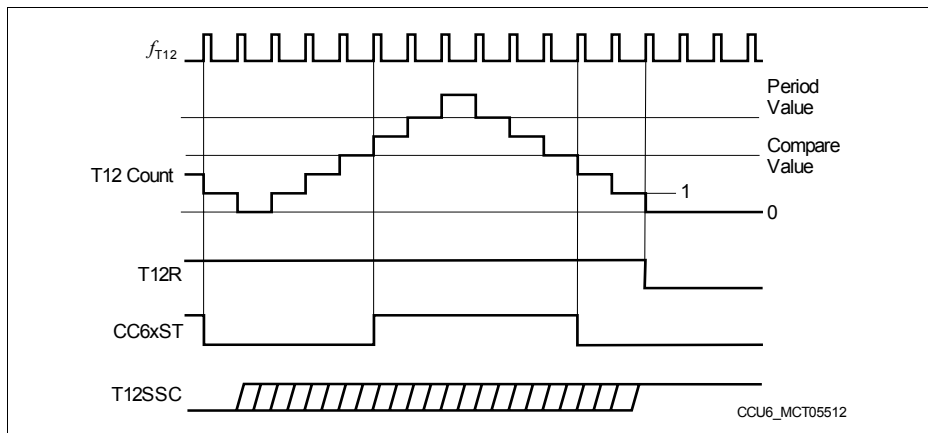


Figure 20-8 Single-Shot Operation in Center-Aligned Mode

Capture/Compare Unit 6 (CCU6)

20.3.3 T12 Compare Mode

Associated with Timer T12 are three individual capture/compare channels, that can perform compare or capture operations with regard to the contents of the T12 counter. The capture functions are explained in [Section 20.3.5](#).

20.3.3.1 Compare Channels

In Compare Mode (see [Figure 20-9](#)), the three individual compare channels CC60, CC61, and CC62 can generate a three-phase PWM pattern.

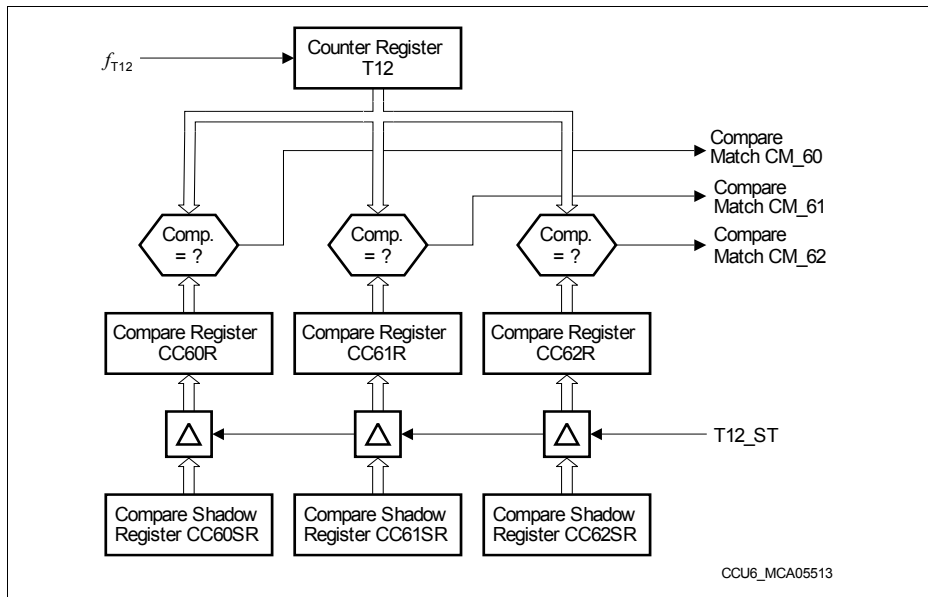


Figure 20-9 T12 Channel Comparators

Each compare channel is connected to the T12 counter register via its individual equal-to comparator, generating a match signal when the contents of the counter matches the contents of the associated compare register. Each channel consists of the comparator and a double register structure - the actual compare register CC6xR, feeding the comparator, and an associated shadow register CC6xSR, that is preloaded by software and transferred into the compare register when signal T12 shadow transfer, T12_ST, gets active. Providing a shadow register for the compare value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters of a three-phase PWM.

Capture/Compare Unit 6 (CCU6)

20.3.3.2 Channel State Bits

Associated with each (compare) channel is a State Bit, **CMPSTATL.CC6xST**, holding the status of the compare (or capture) operation (see [Figure 20-10](#)). In compare mode, the State Bits are modified according to a set of switching rules, depending on the current status of timer T12.

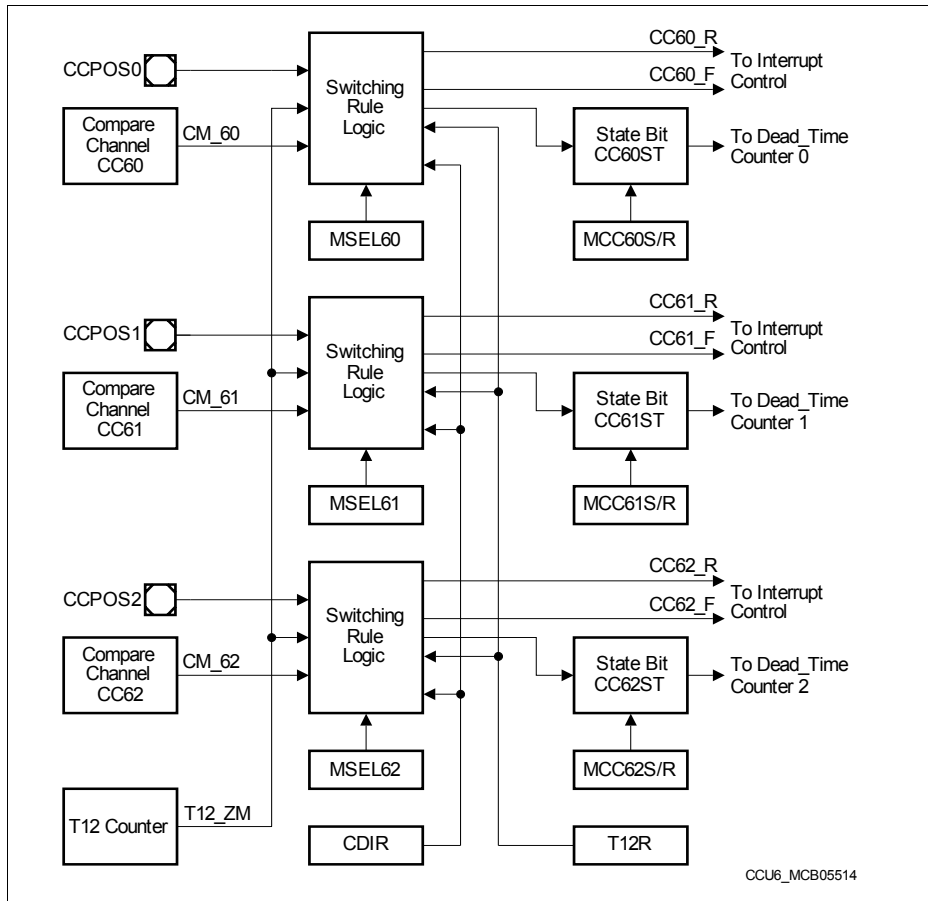


Figure 20-10 Compare State Bits for Compare Mode

The inputs to the switching rule logic for the CC6xST bits are the timer direction (CDIR), the timer run bit (T12R), the timer T12 zero-match signal (T12_ZM), and the actual individual compare-match signals CM_6x as well as the mode control bits, **T12MSEL.MSEL6x**.

Capture/Compare Unit 6 (CCU6)

In addition, each state bit can be set or cleared by software via the appropriate set and reset bits in register **CMPMODIFL**, **CMPMODIFH**, **MCC6xS** and **MCC6xR**. The input signals **CCPOSx** are used in hysteresis-like compare mode, whereas in normal compare mode, these inputs are ignored.

Note: In Hall Sensor, single shot or capture modes, additional/different rules are taken into account (see related sections).

A compare interrupt event **CC6x_R** is signaled when a compare match is detected while counting upwards, whereas the compare interrupt event **CC6x_F** is signaled when a compare match is detected while counting down. The actual setting of a State Bit has no influence on the interrupt generation in compare mode.

A modification of a State Bit **CC6xST** by the switching rule logic due to a compare action is only possible while Timer T12 is running (**T12R** = 1). If this is the case, the following switching rules apply for setting and clearing the State Bits in Compare Mode (illustrated in **Figure 20-11** and **Figure 20-12**):

A State Bit **CC6xST** is set to 1:

- with the next T12 clock (f_{T12}) after a compare-match when T12 is counting up (i.e., when the counter is incremented above the compare value);
- with the next T12 clock (f_{T12}) after a zero-match AND a parallel compare-match when T12 is counting up.

A State Bit **CC6xST** is cleared to 0:

- with the next T12 clock (f_{T12}) after a compare-match when T12 is counting down (i.e., when the counter is decremented below the compare value in center-aligned mode);
- with the next T12 clock (f_{T12}) after a zero-match AND NO parallel compare-match when T12 is counting up.

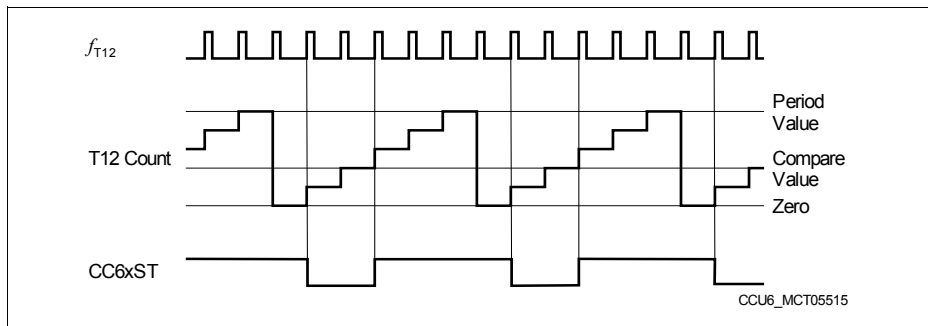


Figure 20-11 Compare Operation, Edge-Aligned Mode

Figure 20-13 illustrates some more examples for compare waveforms. It is important to note that in these examples, it is assumed that some of the compare values are changed

Capture/Compare Unit 6 (CCU6)

while the timer is running. This change is performed via a software preload of the Shadow Register, CC6xSR. The value is transferred to the actual Compare Register CC6xR with the T12 Shadow Transfer signal, T12_ST, that is assumed to be enabled.

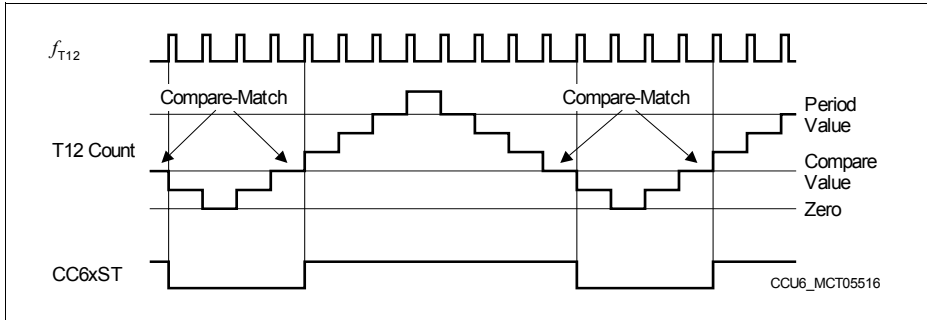


Figure 20-12 Compare Operation, Center-Aligned Mode

Capture/Compare Unit 6 (CCU6)

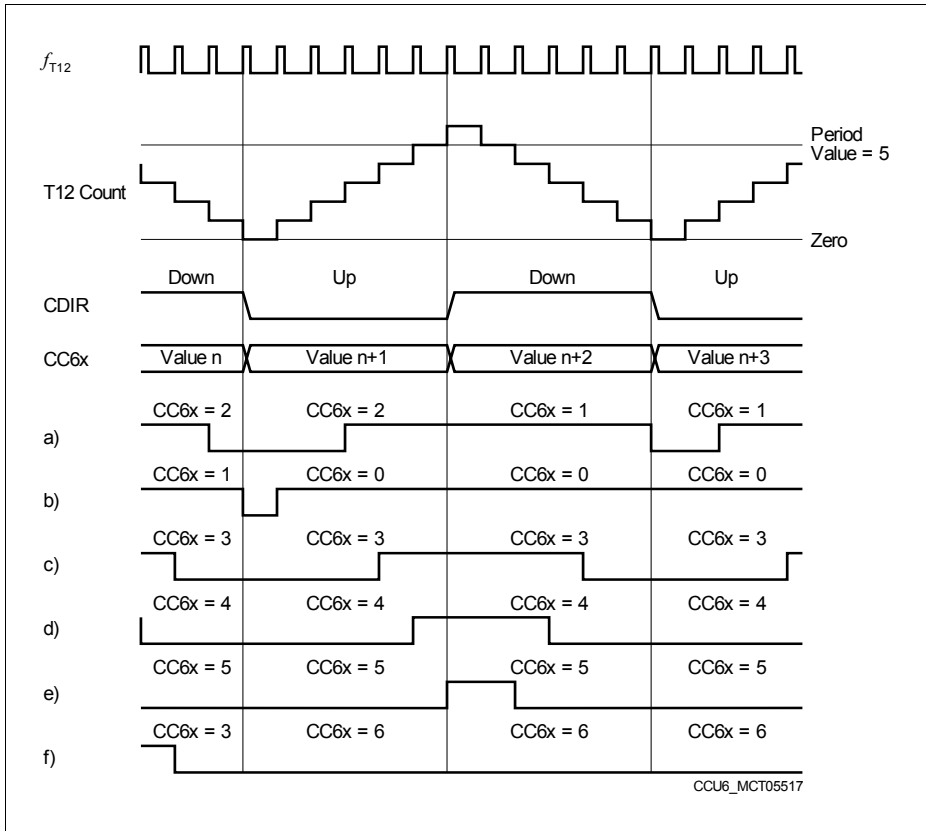


Figure 20-13 Compare Waveform Examples

Example b) illustrates the transition to a duty cycle of 100%. First, a compare value of 0001_H is used, then changed to 0000_H . Please note that a low pulse with the length of one T12 clock is still produced in the cycle where the new value 0000_H is in effect; this pulse originates from the previous value 0001_H . In the following timer cycles, the State Bit CC6xST remains at 1, producing a 100% duty cycle signal. In this case, the compare rule 'zero-match AND compare-match' is in effect.

Example f) shows the transition to a duty cycle of 0%. The new compare value is set to $\langle \text{Period-Value} \rangle + 1$, and the State Bit CC6ST remains cleared.

Figure 20-14 illustrates an example for the waveforms of all three channels. With the appropriate dead-time control and output modulation, a very efficient 3-phase PWM signal can be generated.

Capture/Compare Unit 6 (CCU6)

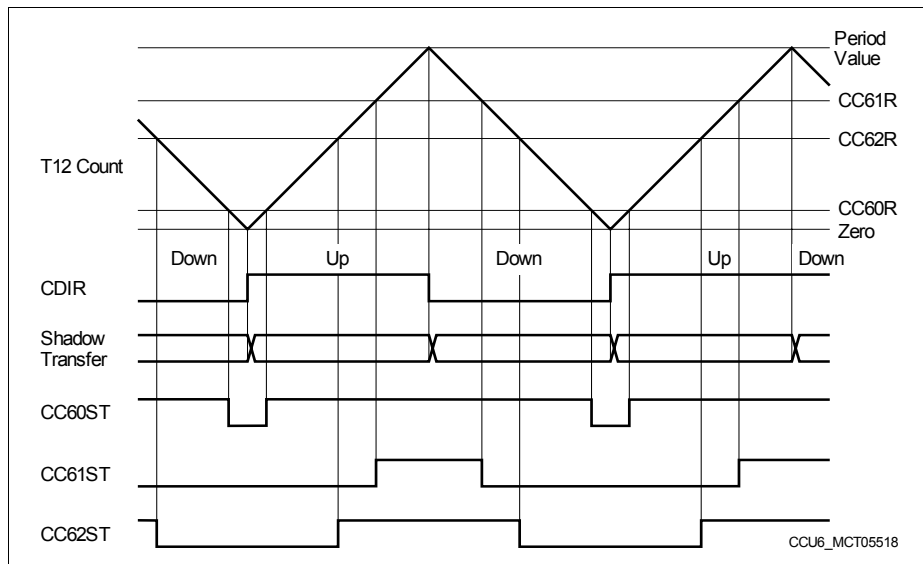


Figure 20-14 Three-Channel Compare Waveforms

20.3.3.3 Hysteresis-Like Control Mode

The hysteresis-like control mode (**T12MSELL**.MSEL6x = 1001_B) offers the possibility to switch off the PWM output if the input CCPOSx becomes 0 by clearing the State Bit CC6xST. This can be used as a simple motor control feature by using a comparator indicating, e.g., overcurrent. While CCPOSx = 0, the PWM outputs of the corresponding channel are driving their passive levels, because the setting of bit CC6xST is only possible while CCPOSx = 1.

As long as input CCPOSx is 0, the corresponding State Bit is held 0. When CCPOSx is at high level, the outputs can be in active state and are determined by bit CC6xST (see **Figure 20-10** for the state bit logic and **Figure 20-15** for the output paths).

The CCPOSx inputs are evaluated with f_{CC6} .

This mode can be used to introduce a timing-related behavior to a hysteresis controller. A standard hysteresis controller detects if a value exceeds a limit and switches its output according to the compare result. Depending on the operating conditions, the switching frequency and the duty cycle are not fixed, but change permanently.

If (outer) time-related control loops based on a hysteresis controller in an inner loop should be implemented, the outer loops show a better behavior if they are synchronized to the inner loops. Therefore, the hysteresis-like mode can be used, that combines timer-related switching with a hysteresis controller behavior. For example, in this mode, an output can be switched on according to a fixed time base, but it is switched off as soon as a falling edge is detected at input CCPOSx.

This mode can also be used for standard PWM with overcurrent protection. As long as there is no low level signal at pin CCPOSx, the output signals are generated in the normal manner as described in the previous sections. Only if input CCPOSx shows a low level, e.g. due to the detection of overcurrent, the outputs are shut off to avoid harmful stress to the system.

20.3.4 Compare Mode Output Path

Figure 20-15 gives an overview on the signal path from a channel State Bit to its output pin in its simplest form. As illustrated, a user has a variety of controls to determine the desired output signal switching behavior in relation to the current state of the State Bit, CC6xST. Please refer to [Section 20.3.4.3](#) for details on the output modulation.

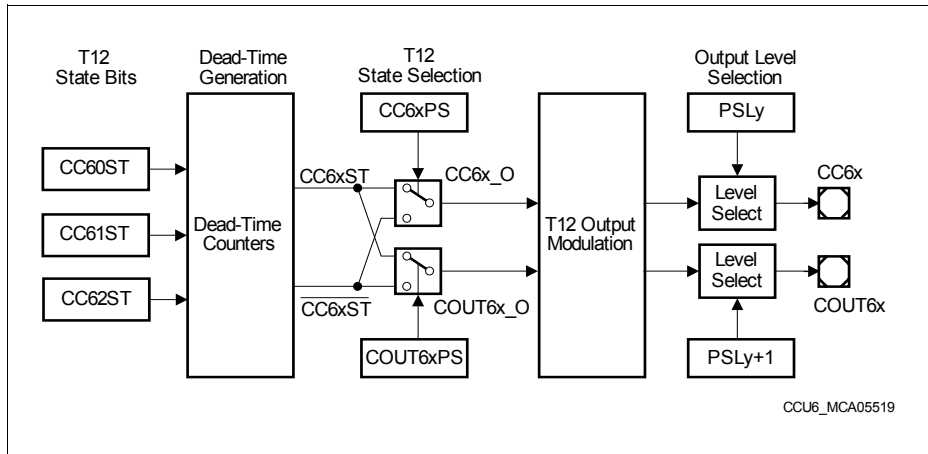


Figure 20-15 Compare Mode Simplified Output Path Diagram

The output path is based on signals that are defined as active or passive. The terms active and passive are not related to output levels, but to internal actions. This mainly applies for the modulation, where T12 and T13 signals are combined with the multi-channel signals and the trap function. The Output level Selection allows the user to define the output level at the output pin for the passive state (inverted level for the active state). It is recommended to configure this block in a way that an external power switch is switched off while the CCU6 delivers an output signal in the passive state.

20.3.4.1 Dead-Time Generation

The generation of (complementary) signals for the high-side and the low-side switches of one power inverter phase is based on the same compare channel. For example, if the high-side switch should be active while the T12 counter value is above the compare value (State Bit = 1), then the low-side switch should be active while the counter value is below the compare value (State Bit = 0).

In most cases, the switching behavior of the connected power switches is not symmetrical concerning the switch-on and switch-off times. A general problem arises if the time for switch-on is smaller than the time for switch-off of the power device. In this case, a short-circuit can occur in the inverter bridge leg, which may damage the complete system. In order to solve this problem by HW, this capture/compare unit

Capture/Compare Unit 6 (CCU6)

contains a programmable Dead-Time Generation Block, that delays the passive to active edge of the switching signals by a programmable time (the active to passive edge is not delayed).

The Dead-Time Generation Block, illustrated in [Figure 20-16](#), is built in a similar way for all three channels of T12. It is controlled by bits in register **T12DTCL**, **T12DTCH**. Any change of a CC6xST State Bit activates the corresponding Dead-Time Counter, that is clocked with the same input clock as T12 (f_{T12}). The length of the dead-time can be programmed by bit field DTM. This value is identical for all three channels. Writing **TCTR4L.DTRES** = 1 sets all dead-times to passive.

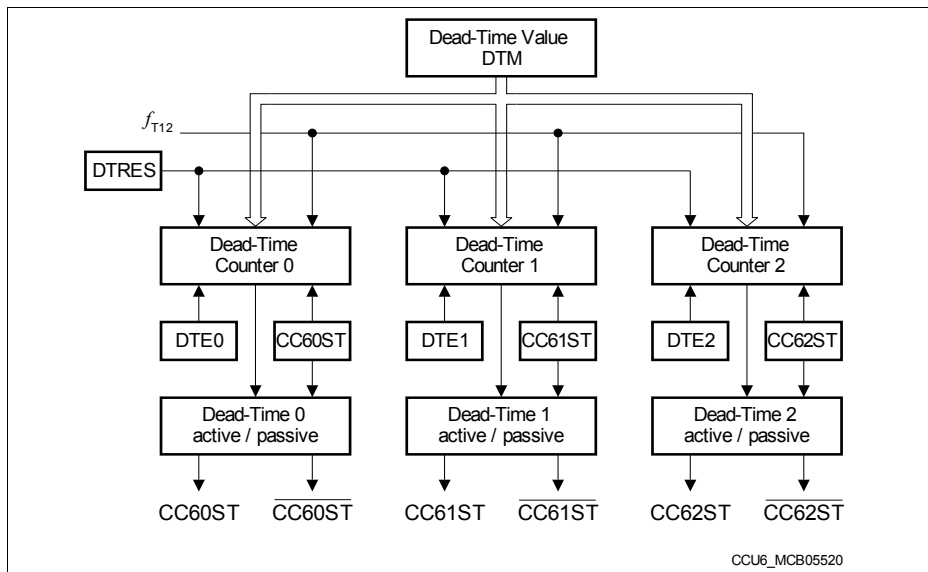


Figure 20-16 Dead-Time Generation Block Diagram

Each of the three dead-time counters has its individual dead-time enable bit, DTE_x. An enabled dead-time counter generates a dead-time delaying the passive-to-active edge of the channel output signal. The change in a State Bit CC6xST is not taken into account while the dead-time generation of this channel is currently in progress (active). This avoids an unintentional additional dead-time if a State Bit CC6xST changes too early. A disabled dead-time counter is always considered as passive and does not delay any edge of CC6xST.

Based on the State Bits CC6xST, the Dead-Time Generation Block outputs a direct signal CC6xST and an inverted signal $\overline{\text{CC6xST}}$ for each compare channel, each masked with the effect of the related Dead-Time Counters (waveforms illustrated in [Figure 20-17](#)).

Capture/Compare Unit 6 (CCU6)

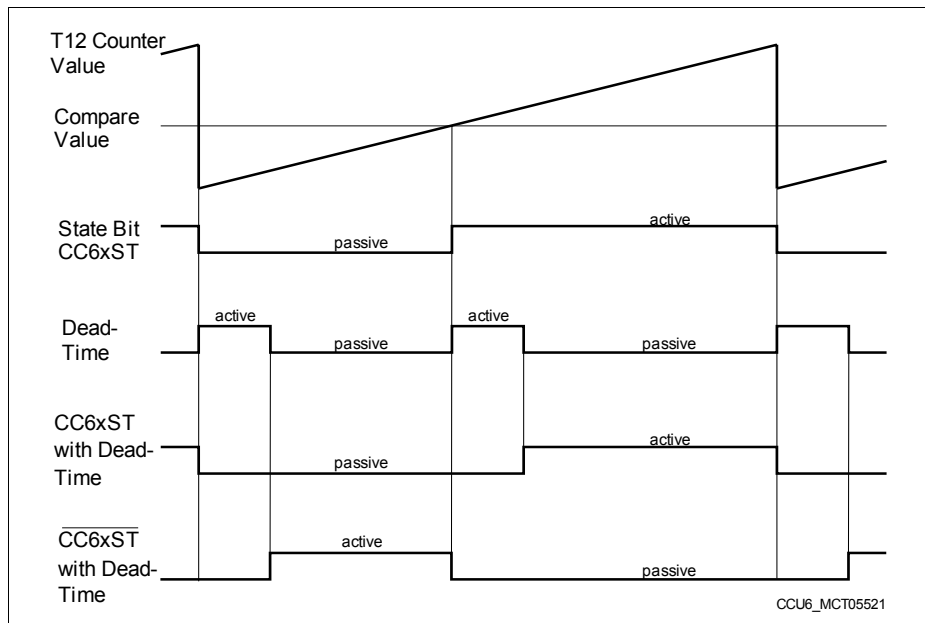


Figure 20-17 Dead-Time Generation Waveforms

20.3.4.2 State Selection

To support a wide range of power switches and drivers, the state selection offers the flexibility to define when an output can be active and can be modulated, especially useful for **complementary or multi-phase PWM** signals.

The state selection is based on the signals CC6xST and CC6xST delivered by the dead-time generator (see [Figure 20-15](#)). Both signals are never active at the same time, but can be passive at the same time. This happens during the dead-time of each compare channel after a change of the corresponding State Bit CC6xST.

The user can select independently for each output signal CC6xO and COUT6xO if it should be active before or after the compare value has been reached (see register [CMPSTATL](#), [CMPSTATH](#)). With this selection, the active (conducting) phases of complementary power switches in a power inverter bridge leg can be positioned with respect to the compare value (e.g. signal CC6xO can be active before, whereas COUT6xO can be active after the compare value is reached). Like this, the output modulation, the trap logic and the output level selection can be programmed independently for each output signal, although two output signals are referring to the same compare channel.

20.3.4.3 Output Modulation and Level Selection

The last block of the data path is the Output Modulation block. Here, all the modulation sources and the trap functionality are combined and control the actual level of the output pins (controlled by the modulation enable bits T1xMODENy and MCMEN in register **MODCTRL**, **MODCTRH**). The following signal sources can be combined here **for each T12 output signal** (see **Figure 20-18** for compare channel CC60):

- A **T12 related compare signal** CC6x_O (for outputs CC6x) or COUT6x_O (for outputs COUT6x) delivered by the T12 block (state selection with dead-time) with an individual enable bit T12MODENy per output signal (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x)
- The **T13 related compare signal** CC63_O delivered by the T13 state selection with an individual enable bit T13MODENy per output signal (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x)
- A **multi-channel output signal** MCMPy (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x) with a common enable bit MCMEN
- The **trap state** TRPS with an individual enable bit TRPENy per output signal (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x)

If one of the modulation input signals CC6x_O/COUT6x_O, CC63_O, or MCMPy of an output modulation block is enabled and is at passive state, the modulated is also in passive state, regardless of the state of the other signals that are enabled. Only if all enabled signals are in active state the modulated output shows an active state. If no modulation input is enabled, the output is in passive state.

If the Trap State is active (TRPS = 1), then the outputs that are enabled for the trap signal (by TRPENy = 1) are set to the passive state.

The output of each of the modulation control blocks is connected to a level select block that is configured by register **PSLR**. It offers the option to determine the actual output level of a pin, depending on the state of the output line (decoupling of active/passive state and output polarity) as specified by the Passive State Select bit PSLy. If the modulated output signal is in the passive state, the level specified directly by PSLy is output. If it is in the active state, the inverted level of PSLy is output. This allows the user to adapt the polarity of an active output signal to the connected circuitry.

The PSLy bits have shadow registers to allow for updates without undesired pulses on the output lines. The bits related to CC6x and COUT6x (x = 0, 1, 2) are updated with the T12 shadow transfer signal (T12_ST). A read action returns the actually used values, whereas a write action targets the shadow bits. Providing a shadow register for the PSL value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters.

Figure 20-18 shows the output modulation structure for compare channel CC60 (output signals CC60 and COUT60). A similar structure is implemented for the other two compare channels CC61 and CC62.

Capture/Compare Unit 6 (CCU6)

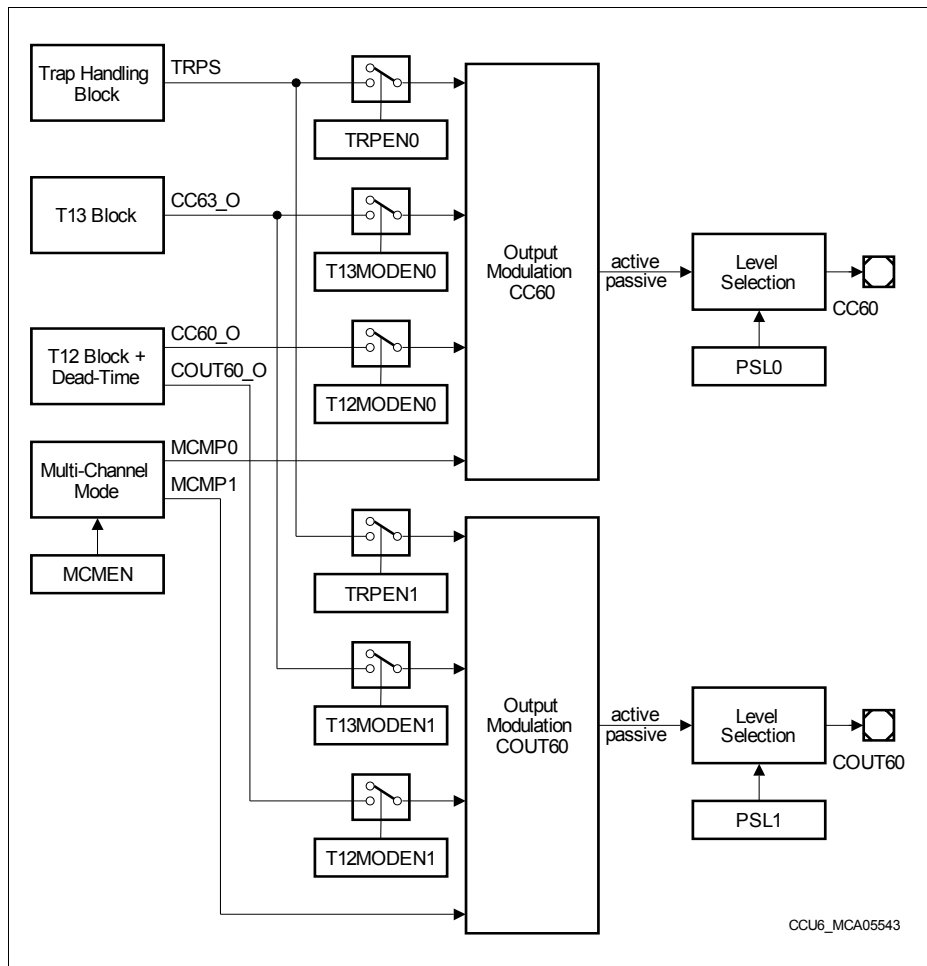


Figure 20-18 Output Modulation for Compare Channel CC60

20.3.5 T12 Capture Modes

Each of the three channels of the T12 Block can also be used to capture T12 time information in response to an external signal CC6xIN.

In capture mode, the interrupt event CC6x_R is detected when a rising edge is detected at the input CC6xIN, whereas the interrupt event CC6x_F is detected when a falling edge is detected.

There are a number of different modes for capture operation. In all modes, both of the registers of a channel are used. The selection of the capture modes is done via the **T12MSEL**.MSEL6x bit fields and can be selected individually for each of the channels.

Table 20-8 Capture Modes Overview

MSEL6x	Mode	Signal	Active Edge	CC6nSR Stored in	T12 Stored in
0100 _B	1	CC6xIN	Rising	–	CC6xR
		CC6xIN	Falling	–	CC6xSR
0101 _B	2	CC6xIN	Rising	CC6xR	CC6xSR
0110 _B	3	CC6xIN	Falling	CC6xR	CC6xSR
0111 _B	4	CC6xIN	Any	CC6xR	CC6xSR

Figure 20-19 illustrates **Capture Mode 1**. When a rising edge (0-to-1 transition) is detected at the corresponding input signal CC6xIN, the current contents of Timer T12 are captured into register CC6xR. When a falling edge (1-to-0 transition) is detected at the input signal CC6xIN, the contents of Timer T12 are captured into register CC6xSR.

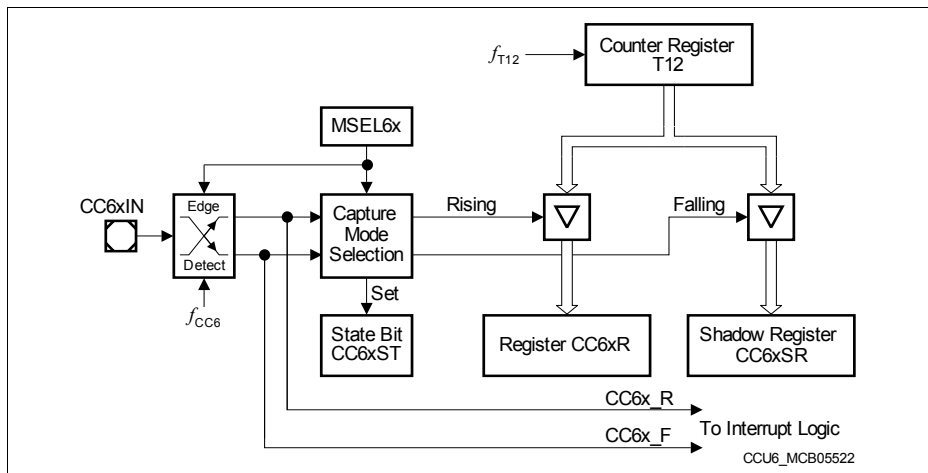


Figure 20-19 Capture Mode 1 Block Diagram

Capture/Compare Unit 6 (CCU6)

Capture Modes 2, 3 and 4 are shown in **Figure 20-20**. They differ only in the active edge causing the capture operation. In each of the three modes, when the selected edge is detected at the corresponding input signal CC6xIN, the current contents of the shadow register CC6xSR are transferred into register CC6xR, and the current Timer T12 contents are captured in register CC6xSR (simultaneous transfer). The active edge is a rising edge of CC6xIN for Capture Mode 2, a falling edge for Mode 3, and both, a rising or a falling edge for Capture Mode 4, as shown in **Table 20-8**. These capture modes are very useful in cases where there is little time between two consecutive edges of the input signal.

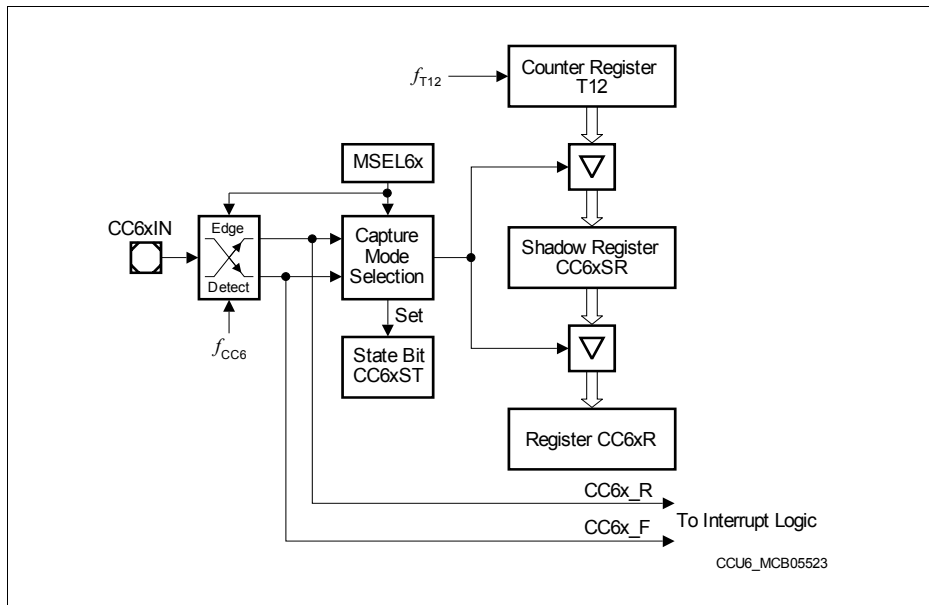


Figure 20-20 Capture Modes 2, 3 and 4 Block Diagram

Capture/Compare Unit 6 (CCU6)

Five further capture modes are called **Multi-Input Capture Modes**, as they use two different external inputs, signal CC6xIN and signal CCPOSx.

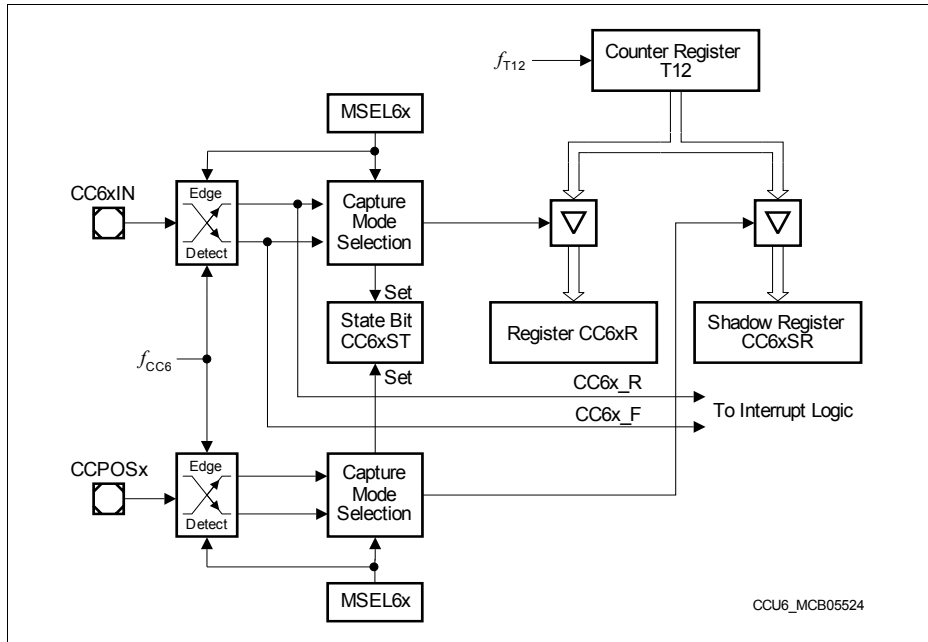


Figure 20-21 Multi-Input Capture Modes Block Diagram

In each of these modes, the current T12 contents are captured in register CC6xR in response to a selected event at signal CC6xIN, and in register CC6xSR in response to a selected event at signal CCPOSx. The possible events can be opposite input transitions, or the same transitions, or any transition at the two inputs. The different options are detailed in [Table 20-9](#).

In each of the various capture modes, the Channel State Bit, CC6xST, is set to 1 when the selected capture trigger event at signal CC6xIN or CCPOSx has occurred. The State Bit is not cleared by hardware, but can be cleared by software.

In addition, appropriate signal lines to the interrupt logic are activated, that can generate an interrupt request to the CPU. Regardless of the selected active edge, all edges detected at signal CC6xIN can lead to the activation of the appropriate interrupt request line (see also [Section 20.9](#)).

Capture/Compare Unit 6 (CCU6)

Table 20-9 Multi-Input Capture Modes Overview

MSEL6x	Mode	Signal	Active Edge	T12 Stored in
1010 _B	5	CC6xIN	Rising	CC6xR
		CCPOSx	Falling	CC6xSR
1011 _B	6	CC6xIN	Falling	CC6xR
		CCPOSx	Rising	CC6xSR
1100 _B	7	CC6xIN	Rising	CC6xR
		CCPOSx	Rising	CC6xSR
1101 _B	8	CC6xIN	Falling	CC6xR
		CCPOSx	Falling	CC6xSR
1110 _B	9	CC6xIN	Any	CC6xR
		CCPOSx	Any	CC6xSR
1111 _B	–	reserved (no capture or compare action)		

20.3.6 T12 Shadow Register Transfer

A special shadow transfer signal (T12_ST) can be generated to facilitate updating the period and compare values of the compare channels CC60, CC61, and CC62 synchronously to the operation of T12. Providing a shadow register for values defining one PWM period facilitates a concurrent update by software for all relevant parameters. The next PWM period can run with a new set of parameters. The generation of this signal is requested by software via bit **TCTR0L.STE12** (set by writing 1 to the write-only bit **TCTR4L.T12STR**, cleared by writing 1 to the write-only bit **TCTR4L.T12STD**).

Figure 20-22 shows the shadow register structure and the shadow transfer signals, as well as on the read/write accessibility of the various registers.

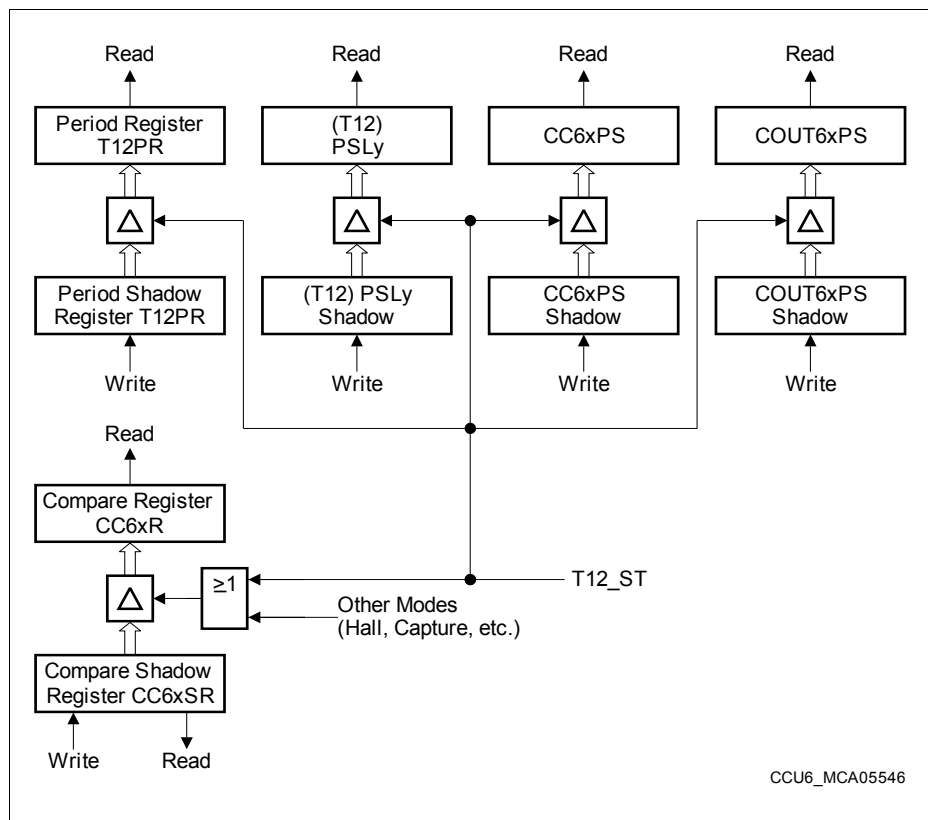


Figure 20-22 T12 Shadow Register Overview

Capture/Compare Unit 6 (CCU6)

A T12 shadow register transfer takes place (T12_ST active):

- while timer T12 is not running (T12R = 0), or
- STE12 = 1 and a Period-Match is detected while counting up, or
- STE12 = 1 and a One-Match is detected while counting down

When signal T12_ST is active, a shadow register transfer is triggered with the next cycle of the T12 clock. Bit STE12 is automatically cleared with the shadow register transfer.

20.3.7 Timer T12 Operating Mode Selection

The operating mode for the T12 channels are defined by the bit fields **TCTR4L.MSEL6x**, **T12MSELL.MSEL6x**.

Table 20-10 T12 Capture/Compare Modes Overview

MSEL6x	Selected Operating Mode
0000 _B , 1111 _B	Capture/Compare modes switched off
0001 _B , 0010 _B , 0011 _B	Compare mode, see Section 20.3.3 same behavior for all three codings
01XX _B	Double-Register Capture modes, see Section 20.3.5
1000 _B	Hall Sensor Mode, see Section 20.7 In order to properly enable this mode, all three MSEL6x fields have to be programmed to Hall Sensor mode.
1001 _B	Hysteresis-like compare mode, see Section 20.3.3.3
1010 _B , 1011 _B , 1100 _B , 1101 _B , 1110 _B	Multi-Input Capture modes, see Section 20.3.5

The clocking and counting scheme of the timers are controlled by the timer control registers **TCTR0L**, **TCTR0H** and **TCTR2L**, **TCTR2H**. Specific actions are triggered by write operations to register **TCTR4L**, **TCTR4H**.

20.3.8 T12 related Registers

20.3.8.1 T12 Counter Register

Register T12 represents the counting value of timer T12. It can only be written while the timer T12 is stopped. Write actions while T12 is running are not taken into account. Register T12 can always be read by SW.

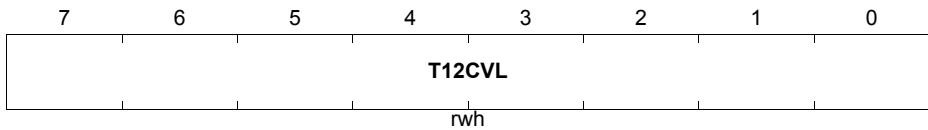
In edge-aligned mode, T12 only counts up, whereas in center-aligned mode, T12 can count up and down.

T12L

Timer T12 Counter Register Low (FA_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3



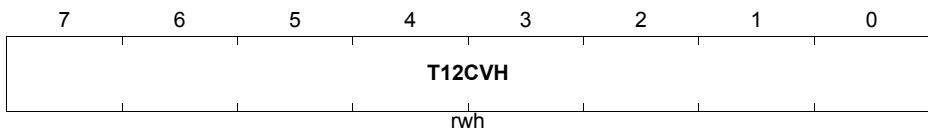
Field	Bits	Type	Description
T12CVL	[7:0]	rwh	Timer T12 Counter Value This register represents lower 8-bits of the 16-bit counter value of T12.

T12H

Timer T12 Counter Register High (FB_H)

Reset Value: 00_H

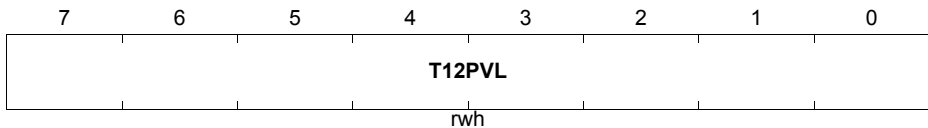
RMAP: 0, PAGE: 3



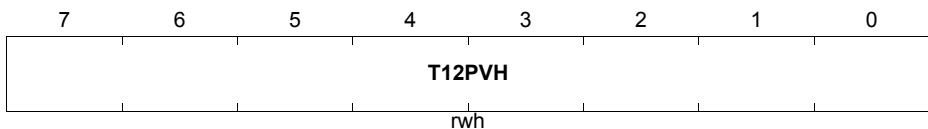
Field	Bits	Type	Description
T12CVH	[7:0]	rwh	Timer T12 Counter Value This register represents upper 8-bits of the 16-bit counter value of T12.

Capture/Compare Unit 6 (CCU6)
20.3.8.2 Period Register

Register T12PRL/H contains the period value for timer T12. The period value is compared to the actual counter value of T12 and the resulting counter actions depend on the defined counting rules. This register has a shadow register and the shadow transfer is controlled by bit STE12. A read action by SW delivers the value that is currently used for the compare action, whereas the write action targets a shadow register. The shadow register structure allows a concurrent update of all T12-related values.

T12PRL
Timer T12 Period Register Low
(9C_H)
Reset Value: 00_H
RMAP: 0, PAGE: 1


Field	Bits	Type	Description
T12PVL	[7:0]	rwh	Timer T12 Period Value The value T12PVL defines lower 8-bits of the counter value for T12 leading to a period-match. When reaching this value, the timerT12 is set to zero (edge-aligned mode) or changes its count direction to down counting (center-aligned mode).

T12PRH
Timer T12 Period Register High
(9D_H)
Reset Value: 00_H
RMAP: 0, PAGE: 1


Capture/Compare Unit 6 (CCU6)

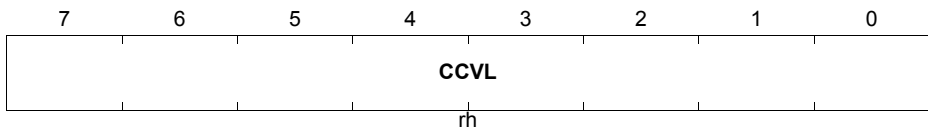
Field	Bits	Type	Description
T12PVH	[7:0]	rwh	Timer T12 Period Value The value T12PVL defines the upper 8-bits of counter value for T12 leading to a period-match. When reaching this value, the timerT12 is set to zero (edge-aligned mode) or changes its count direction to down counting (center-aligned mode).

Capture/Compare Unit 6 (CCU6)
20.3.8.3 Capture/Compare Registers

In compare mode, the registers CC6xRL/H ($x = 0 - 2$) are the actual compare registers for T12. The values stored in CC6xR are compared (all three channels in parallel) to the counter value of T12. In capture mode, the current value of the T12 counter register is captured by registers CC6xRL/H if the corresponding capture event is detected.

CC6xRL ($x = 0-2$)
Capture/Compare Register for Channel CC6x Low

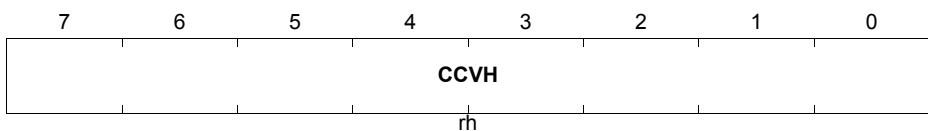
$$(FA_H + x * 2)$$

Reset Value: 00_H
RMAP: 0, PAGE: 1


Field	Bits	Type	Description
CCVL	[7:0]	rh	Capture/Compare Value In compare mode, the bit fields CCV contain the values, that are compared to the T12 counter value. In capture mode, the captured value of T12 can be read from these registers.

CC6xRH ($x = 0-2$)
Capture/Compare Register for Channel CC6x High

$$(FB_H + x * 2)$$

Reset Value: 00_H
RMAP: 0, PAGE: 1


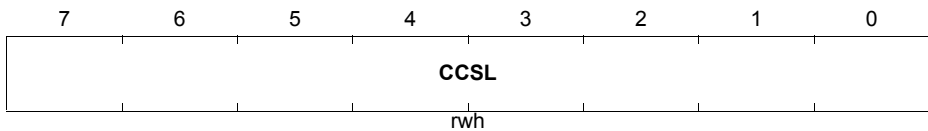
Field	Bits	Type	Description
CCVH	[7:0]	rh	Capture/Compare Value In compare mode, the bit fields CCV contain the values, that are compared to the T12 counter value. In capture mode, the captured value of T12 can be read from these registers.

Capture/Compare Unit 6 (CCU6)
20.3.8.4 Capture/Compare Shadow Registers

The registers CC6xRL/H can only be read by SW, the modification of the value is done by a shadow register transfer from register CC6xSRL/H. The corresponding shadow registers CC6xSRL/H can be read and written by SW. In capture mode, the value of the T12 counter register can also be captured by registers CC6xSRL/H if the selected capture event is detected (depending on the selected capture mode).

CC6xSRL (x=0-2)
Capture/Compare Shadow Register for Channel CC6x Low

$$(FA_H + x * 2)$$

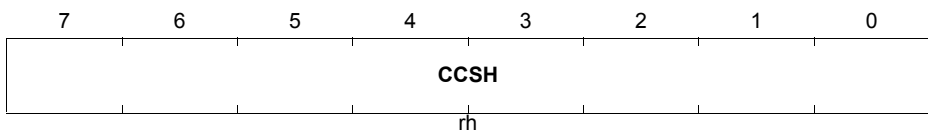
Reset Value: 00_H
RMAP: 0, PAGE: 0


Field	Bits	Type	Description
CCSL	[7:0]	rh	Shadow Register for Channel x Capture/Compare Value In compare mode, the bit fields contents of CCS are transferred to the bit fields CCV for the corresponding channel during a shadow transfer. In capture mode, the captured value of T12 can be read from these registers.

Note: The shadow registers can also be written by SW in capture mode. In this case, the HW capture event wins over the SW write if both happen in the same cycle (the SW write is discarded).

CC6xSRH (x=0-2)
Capture/Compare Shadow Register for Channel CC6x High

$$(FB_H + x * 2)$$

Reset Value: 00_H
RMAP: 0, PAGE: 0


Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
CCSH	[7:0]	rh	Shadow Register for Channel x Capture/Compare Value In compare mode, the bit fields contents of CCS are transferred to the bit fields CCV for the corresponding channel during a shadow transfer. In capture mode, the captured value of T12 can be read from these registers.

Note: The shadow registers can also be written by SW in capture mode. In this case, the HW capture event wins over the SW write if both happen in the same cycle (the SW write is discarded).

20.3.8.5 Dead-time Control Register

Register T12DTCL/H controls the dead-time generation for the timer T12 compare channels. Each channel can be independently enabled/disabled for dead-time generation. If enabled, the transition from passive state to active state is delayed by the value defined by bit field DTM.

The dead time counters are clocked with the same frequency as T12.

This structure allows symmetrical dead-time generation in center-aligned and in edge-aligned PWM mode. A duty cycle of 50% leads to CC6x, COUT6x switched on for: 0.5 * period - dead time.

Note: The dead-time counters are not reset by bit T12RES, but by bit DTRES.

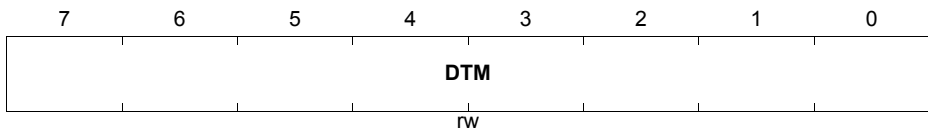
T12DTCL

Dead-Time Control Register for Timer T12 Low

(A4_H)

Reset Value: 00_H

RMAP: 0, PAGE: 1



Field	Bits	Type	Description
DTM	[7:0]	rw	Dead-Time Bit field DTM determines the programmable delay between switching from the passive state to the active state of the selected outputs. The switching from the active state to the passive state is not delayed.

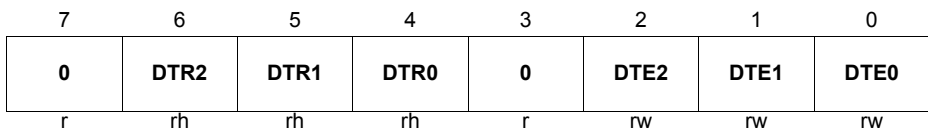
T12DTCH

Dead-Time Control Register for Timer T12 High

(A5_H)

Reset Value: 00_H

RMAP: 0, PAGE: 1



Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
DTE2, DTE1, DTE0	2, 1, 0	rw	Dead Time Enable Bits Bits DTE0..DTE2 enable and disable the dead time generation for each compare channel (0, 1, 2) of timer T12. 0 _B Dead-Time Counter x is disabled. The corresponding outputs switch from the passive state to the active state (according to the actual compare status) without any delay. 1 _B Dead-Time Counter x is enabled. The corresponding outputs switch from the passive state to the active state (according to the compare status) with the delay programmed in bit field DTM.
DTR2, DTR1, DTR0	6, 5, 4	rw	Dead Time Run Indication Bits Bits DTR0..DTR2 indicate the status of the dead time generation for each compare channel (0, 1, 2) of timer T12. 0 _B Dead-Time Counter x is currently in the passive state. 1 _B Dead-Time Counter x is currently in the active state.
0	7, 3	r	reserved; returns 0 if read; should be written with 0;

Capture/Compare Unit 6 (CCU6)

20.3.9 Capture/Compare Control Registers

20.3.9.1 Channel State Bits

The Compare State Register CMPSTATL/H contains status bits monitoring the current capture and compare state and control bits defining the active/passive state of the compare channels.

CMPSTATL

Compare State Register Low

(FE_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	CC63ST	CCPOS2	CCPOS1	CCPOS0	CC62ST	CC61ST	CC60ST
r	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
CC60ST, CC61ST, CC62ST, CC63ST ¹⁾	0, 1, 2, 6	rh	Capture/Compare State Bits Bits CC6xST monitor the state of the capture/compare channels. Bits CC6xST (x = 0, 1, 2) are related to T12, bit CC63ST is related to T13. 0 _B In compare mode, the timer count is less than the compare value. In capture mode, the selected edge has not yet been detected since the bit has been cleared by SW the last time. 1 _B In compare mode, the counter value is greater than or equal to the compare value. In capture mode, the selected edge has been detected.
CCPOS60, CCPOS61, CCPOS62	3, 4, 5	rh	Sampled Hall Pattern Bits Bits CCPOS6x (x = 0, 1, 2) are indicating the value of the input Hall pattern that has been compared to the current and expected value. The value is sampled when the event HCRDY (Hall Compare Ready) occurs. 0 _B The input CCPOS6x has been sampled as 0. 1 _B The input CCPOS6x has been sampled as 1.
0	7	r	reserved; returns 0 if read; should be written with 0;

Capture/Compare Unit 6 (CCU6)

1) These bits are set and cleared according to the T12, T13 switching rules

CMPSTATH

Compare State Register High

(FF_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
T13IM	COUT63PS	COUT62PS	CC62PS	COUT61PS	CC61PS	COUT60PS	CC60PS
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CC60PS, CC61PS, CC62PS, COUT60PS, COUT61PS, COUT62PS, COUT63PS 1)	0, 2, 4, 1, 3, 5, 6	rwh	Passive State Select for Compare Outputs Bits CC6xPS, COUT6xPS select the state of the corresponding compare channel, that is considered to be the passive state. During the passive state, the passive level (defined in register PSLR) is driven by the output pin. Bits CC6xPS, COUT6xPS (x = 0, 1, 2) are related to T12, bit CC63PS is related to T13. 0 _B The corresponding compare signal is in passive state while CC6xST is 0. 1 _B The corresponding compare signal is in passive state while CC6xST is 1. In capture mode, these bits are not used.
T13IM ²⁾	15	rwh	T13 Inverted Modulation Bit T13IM inverts the T13 signal for the modulation of the CC6x and COUT6x (x = 0, 1, 2) signals. 0 _B T13 output CC63_O is equal to <u>CC63ST</u> . 1 _B T13 output CC63_O is equal to <u>CC63ST</u> .

1) These bits have shadow bits and are updated in parallel to the capture/compare registers of T12, T13 respectively. A read action targets the actually used values, whereas a write action targets the shadow bits.

2) This bit has a shadow bit and is updated in parallel to the compare and period registers of T13. A read action targets the actually used values, whereas a write action targets the shadow bit.

Capture/Compare Unit 6 (CCU6)

The Compare Status Modification Register CMPMODIFL/H provides software-control (independent set and clear conditions) for the channel state bits CC6xST. This feature enables the user to individually change the status of the output lines by software, for example when the corresponding compare timer is stopped.

CMPMODIFL

Compare State Modification Register Low

(A6_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	MCC63S		0		MCC62S	MCC61S	MCC60S
r	w		r		w	w	w

Field	Bits	Type	Description
MCC60S, MCC61S, MCC62S, MCC63S	0, 1, 2, 6	w	Capture/Compare Status Modification Bits These bits are used to bits to set (MCC6xS) or to clear (MCC6xR ¹⁾) the corresponding bits CC6xST by SW. This feature allows the user to individually change the status of the output lines by SW, e.g. when the corresponding compare timer is stopped. This allows a bit manipulation of CC6xST-bits by a single data write action. The following functionality of a write access to bits concerning the same capture/compare state bit is provided: [MCC6xR, MCC6xS] = 00 _B Bit CC6xST is not changed. 01 _B Bit CC6xST is set. 10 _B Bit CC6xST is cleared. 11 _B reserved
0	[5:3], 7	r	reserved; returns 0 if read; should be written with 0;

1) This bit field is contained in the Compare State Modification Register High

Capture/Compare Unit 6 (CCU6)

CMPMODIFH

Compare State Modification Register High

(A7_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	MCC63R		0		MCC62R	MCC61R	MCC60R
r	w		0		rw	w	w

Field	Bits	Type	Description
MCC60R, MCC61R, MCC62R, MCC63R	0, 1, 2, 6	w	Capture/Compare Status Modification Bits These bits are used to bits to set (MCC6xS ¹⁾) or to clear (MCC6xR) the corresponding bits CC6xST by SW. This feature allows the user to individually change the status of the output lines by SW, e.g. when the corresponding compare timer is stopped. This allows a bit manipulation of CC6xST-bits by a single data write action. The following functionality of a write access to bits concerning the same capture/compare state bit is provided: [MCC6xR, MCC6xS] = 00 _B Bit CC6xST is not changed. 01 _B Bit CC6xST is set. 10 _B Bit CC6xST is cleared. 11 _B reserved
0	[5:3], 7	r	reserved; returns 0 if read; should be written with 0;

1) This bit field is contained in the Compare State Modification Register Low

Capture/Compare Unit 6 (CCU6)

20.3.9.2 T12 Mode Control Register

Register T12MSELL/H contains control bits to select the capture/compare functionality of the three channels of Timer T12.

T12MSELL

T12 Mode Select Register Low

(9A_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
MSEL61				MSEL60			
rw				rw			

Field	Bits	Type	Description
MSEL60, MSEL61	[3:0], [7:4]	rw	Capture/Compare Mode Selection These bit fields select the operating mode of the three T12 capture/compare channels. Each channel (x = 0, 1, 2) can be programmed individually for one of these modes (except for Hall Sensor Mode). Coding see Table 20-10 .

T12MSELH

T12 Mode Select Register High

(9B_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
DBYP	HSYNC			MSEL62			
rw	rw			rw			

Field	Bits	Type	Description
MSEL62	[3:0]	rw	Capture/Compare Mode Selection These bit fields select the operating mode of the three T12 capture/compare channels. Each channel (x = 0, 1, 2) can be programmed individually for one of these modes (except for Hall Sensor Mode). Coding see Table 20-10 .

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
HSYNC	[6:4]	rw	Hall Synchronization Bit field HSYNC defines the source for the sampling of the Hall input pattern and the comparison to the current and the expected Hall pattern bit fields. Coding see Table 20-15 .
DBYP	7	rw	Delay Bypass DBYP controls whether the source signal for the sampling of the Hall input pattern (selected by HSYNC) is delayed by the Dead-Time Counter 0. 0_B The bypass is not active. Dead-Time Counter 0 is generating a delay after the source signal becomes active. 1_B The bypass is active. Dead-Time Counter 0 is not used for a delay.

Capture/Compare Unit 6 (CCU6)

20.3.9.3 Timer Control Registers

Register TCTR0L/H controls the basic functionality of both timers, T12 and T13.

Note: A write action to the bit fields T12CLK or T12PRE is only taken into account while the timer T12 is not running (T12R=0). A write action to the bit fields T13CLK or T13PRE is only taken into account while the timer T13 is not running (T13R=0).

TCTR0L

Timer Control Register 0 Low

(A6_H)

Reset Value: 00_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
CTM	CDIR	STE12	T12R	T12PRE	T12CLK		
rw	rh	rh	rh	rw	rw		

Field	Bits	Type	Description
T12CLK	[2:0]	rw	Timer T12 Input Clock Select Selects the input clock for timer T12 that is derived from the peripheral clock according to the equation $f_{T12} = f_{CC6} / 2^{<T12CLK>}$. 000 _B $f_{T12} = f_{CC6}$ 001 _B $f_{T12} = f_{CC6} / 2$ 010 _B $f_{T12} = f_{CC6} / 4$ 011 _B $f_{T12} = f_{CC6} / 8$ 100 _B $f_{T12} = f_{CC6} / 16$ 101 _B $f_{T12} = f_{CC6} / 32$ 110 _B $f_{T12} = f_{CC6} / 64$ 111 _B $f_{T12} = f_{CC6} / 128$
T12PRE	3	rw	Timer T12 Prescaler Bit In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T12. 0 _B The additional prescaler for T12 is disabled. 1 _B The additional prescaler for T12 is enabled.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
T12R	4	rh	Timer T12 Run Bit¹⁾ T12R starts and stops timer T12. It is set/cleared by SW by setting bits T12RR or T12RS or it is cleared by HW according to the function defined by bit field T12SSC. 0 _B Timer T12 is stopped. 1 _B Timer T12 is running.
STE12	5	rh	Timer T12 Shadow Transfer Enable Bit STE12 enables or disables the shadow transfer of the T12 period value, the compare values and passive state select bits and levels from their shadow registers to the actual registers if a T12 shadow transfer event is detected. Bit STE12 is cleared by hardware after the shadow transfer. A T12 shadow transfer event is a period-match while counting up or a one-match while counting down. 0 _B The shadow register transfer is disabled. 1 _B The shadow register transfer is enabled.
CDIR	6	rh	Count Direction of Timer T12 This bit is set/cleared according to the counting rules of T12. 0 _B T12 counts up. 1 _B T12 counts down.
CTM	7	rw	T12 Operating Mode 0 _B Edge-aligned Mode: T12 always counts up and continues counting from zero after reaching the period value. 1 _B Center-aligned Mode: T12 counts down after detecting a period-match and counts up after detecting a one-match.

1) A concurrent set/clear action on T12R (from T12SSC, T12RR or T12RS) will have no effect. The bit T12R will remain unchanged.

Capture/Compare Unit 6 (CCU6)

TCTR0H

Timer Control Register 0 High

(A7_H)

Reset Value: 00_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	STE13	T13R	T13PRE			T13CLK	
r	rh	rh	rw			rw	

Field	Bits	Type	Description
T13CLK	[2:0]	rw	Timer T13 Input Clock Select Selects the input clock for timer T13 that is derived from the peripheral clock according to the equation $f_{T13} = f_{CC6} / 2^{<T13CLK>}$. 000 _B $f_{T13} = f_{CC6}$ 001 _B $f_{T13} = f_{CC6} / 2$ 010 _B $f_{T13} = f_{CC6} / 4$ 011 _B $f_{T13} = f_{CC6} / 8$ 100 _B $f_{T13} = f_{CC6} / 16$ 101 _B $f_{T13} = f_{CC6} / 32$ 110 _B $f_{T13} = f_{CC6} / 64$ 111 _B $f_{T13} = f_{CC6} / 128$
T13PRE	3	rw	Timer T13 Prescaler Bit In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T13. 0 _B The additional prescaler for T13 is disabled. 1 _B The additional prescaler for T13 is enabled.
T13R	4	rh	Timer T13 Run Bit⁽¹⁾ T13R starts and stops timer T13. It is set/cleared by SW by setting bits T13RR or T13RS or it is set/cleared by HW according to the function defined by bit fields T13SSC, T13TEC and T13TED. 0 _B Timer T13 is stopped. 1 _B Timer T13 is running.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
STE13	5	rh	Timer T13 Shadow Transfer Enable Bit STE13 enables or disables the shadow transfer of the T13 period value, the compare value and passive state select bit and level from their shadow registers to the actual registers if a T13 shadow transfer event is detected. Bit STE13 is cleared by hardware after the shadow transfer. A T13 shadow transfer event is a period-match. 0 _B The shadow register transfer is disabled. 1 _B The shadow register transfer is enabled.
0	[7: 6]	r	reserved; returns 0 if read; should be written with 0;

1) A concurrent set/cleared action on T13R (from T13SSC, T13TEC, T13RR or T13RS) will have no effect. The bit T12R will remain unchanged.

Capture/Compare Unit 6 (CCU6)

Register TCTR2L/H controls the single-shot and the synchronization functionality of both timers T12 and T13. Both timers can run in single-shot mode. In this mode they stop their counting sequence automatically after one counting period with a count value of zero. The single-shot mode and the synchronization feature of T13 to T12 allow the generation of events with a programmable delay after well-defined PWM actions of T12.

TCTR2L

Timer Control Register 2 Low
RMAP: 0, PAGE: 2

(FA_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
0	T13TED		T13TEC			T13SSC	T12SSC
r	rw		rw			rw	rw

Field	Bits	Type	Description
T12SSC	0	rw	Timer T12 Single Shot Control This bit controls the single shot-mode of T12. 0 _B The single-shot mode is disabled, no HW action on T12R. 1 _B The single shot mode is enabled, the bit T12R is cleared by HW if - T12 reaches its period value in edge-aligned mode - T12 reaches the value 1 while down counting in center-aligned mode. In parallel to the clear action of bit T12R, the bits CC6xST (x=0, 1, 2) are cleared.
T13SSC	1	rw	Timer T13 Single Shot Control This bit controls the single shot-mode of T13. 0 _B No HW action on T13R 1 _B The single-shot mode is enabled, the bit T13R is cleared by HW if T13 reaches its period value. In parallel to the clear action of bit T13R, the bit CC63ST is cleared.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
T13TEC	[4:2]	rw	T13 Trigger Event Control bit field T13TEC selects the trigger event to start T13 (automatic set of T13R for synchronization to T12 compare signals) according to following combinations: 000 _B no action 001 _B set T13R on a T12 compare event on channel 0 010 _B set T13R on a T12 compare event on channel 1 011 _B set T13R on a T12 compare event on channel 2 100 _B set T13R on any T12 compare event (ch. 0, 1, 2) 101 _B set T13R upon a period-match of T12 110 _B set T13R upon a zero-match of T12 (while counting up) 111 _B set T13R on any edge of inputs CCPOSx
T13TED	[6:5]	rw	Timer T13 Trigger Event Direction¹⁾ Bit field T13TED delivers additional information to control the automatic set of bit T13R in the case that the trigger action defined by T13TEC is detected. 00 _B reserved, no action 01 _B while T12 is counting up 10 _B while T12 is counting down 11 _B independent on the count direction of T12
0	7	r	reserved; returns 0 if read; should be written with 0;

1) Example:

If the timer T13 is intended to start at any compare event on T12 (T13TEC=100) the trigger event direction can be programmed to

- counting up >> a T12 channel 0, 1, 2 compare match triggers T13R only while T12 is counting up
- counting down >> a T12 channel 0, 1, 2 compare match triggers T13R only while T12 is counting down
- independent from bit CDIR >> each T12 channel 0, 1, 2 compare match triggers T13R

The timer count direction is taken from the value of bit CDIR. As a result, if T12 is running in edge-aligned mode (counting up only), T13 can only be started automatically if bit field T13TED=01 or 11.

Capture/Compare Unit 6 (CCU6)

TCTR2H

Timer Control Register 2 High

(FB_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0				T13RSEL		T12RSEL	
r				rw		rw	

Field	Bits	Type	Description
T12RSEL	[1:0]	rw	Timer T12 External Run Selection Bit field T12RSEL defines the event of signal T12HR that can set the run bit T12R by HW. 00 _B The external setting of T12R is disabled. 01 _B Bit T12R is set if a rising edge of signal T12HR is detected. 10 _B Bit T12R is set if a falling edge of signal T12HR is detected. 11 _B Bit T12R is set if an edge of signal T12HR is detected.
T13RSEL	[3:2]	rw	Timer T13 External Run Selection Bit field T13RSEL defines the event of signal T13HR that can set the run bit T13R by HW. 00 _B The external setting of T13R is disabled. 01 _B Bit T13R is set if a rising edge of signal T13HR is detected. 10 _B Bit T13R is set if a falling edge of signal T13HR is detected. 11 _B Bit T13R is set if an edge of signal T13HR is detected.
0	[7:4]	r	reserved; returns 0 if read; should be written with 0;

Capture/Compare Unit 6 (CCU6)

Register TCTR4L/H provides software-control (independent set and clear conditions) for the run bits T12R and T13R. Furthermore, the timers can be reset (while running) and bits STE12 and STE13 can be controlled by software. Reading these bits always returns 0.

TCTR4L

Timer Control Register 4 Low

(9C_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
T12STD	T12STR	0		DTRES	T12RES	T12RS	T12RR
w	w	r		w	w	w	w

Field	Bits	Type	Description
T12RR	0	w	Timer T12 Run Reset Setting this bit clears the T12R bit. 0 _B T12R is not influenced. 1 _B T12R is cleared, T12 stops counting.
T12RS	1	w	Timer T12 Run Set Setting this bit sets the T12R bit. 0 _B T12R is not influenced. 1 _B T12R is set, T12 starts counting.
T12RES	2	w	Timer T12 Reset 0 _B No effect on T12. 1 _B The T12 counter register is cleared to zero. The switching of the output signals is according to the switching rules. Setting of T12RES has no impact on bit T12R.
DTRES	3	w	Dead-Time Counter Reset 0 _B No effect on the dead-time counters. 1 _B The three dead-time counter channels are cleared to zero.
T12STR	6	w	Timer T12 Shadow Transfer Request 0 _B No action 1 _B STE12 is set, enabling the shadow transfer.
T12STD	7	w	Timer T12 Shadow Transfer Disable 0 _B No action 1 _B STE12 is cleared without triggering the shadow transfer.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
0	[5:4]	r	reserved; returns 0 if read; should be written with 0;

Note: A simultaneous write of a 1 to bits that set and clear the same bit will trigger no action. The corresponding bit will remain unchanged.

TCTR4H

Timer Control Register 4 High

(9D_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
T13STD	T13STR		0		T13RES	T13RS	T13RR
w	w		r		w	w	w

Field	Bits	Type	Description
T13RR	0	w	Timer T13 Run Reset Setting this bit clears the T13R bit. 0 _B T13R is not influenced. 1 _B T13R is cleared, T13 stops counting.
T13RS	1	w	Timer T13 Run Set Setting this bit sets the T13R bit. 0 _B T13R is not influenced. 1 _B T13R is set, T13 starts counting.
T13RES	2	w	Timer T13 Reset 0 _B No effect on T13. 1 _B The T13 counter register is cleared to zero. The switching of the output signals is according to the switching rules. Setting of T13RES has no impact on bit T13R.
T13STR	6	w	Timer T13 Shadow Transfer Request 0 _B No action 1 _B STE13 is set, enabling the shadow transfer.
T13STD	7	w	Timer T13 Shadow Transfer Disable 0 _B No action 1 _B STE13 is cleared without triggering the shadow transfer.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
0	[5:3]	r	reserved; returns 0 if read; should be written with 0;

Note: A simultaneous write of a 1 to bits that set and clear the same bit will trigger no action. The corresponding bit will remain unchanged.

20.4 Operating Timer T13

Timer T13 is implemented similarly to Timer T12, but only with one channel in compare mode. A 16-bit up-counter is connected to a channel register via a comparator, that generates a signal when the counter contents match the contents of the channel register. A variety of control functions facilitate the adaptation of the T13 structure to different application needs. In addition, T13 can be started synchronously to timer T12 events.

This section provides information about:

- T13 overview (see [Section 20.4.1](#))
- Counting scheme (see [Section 20.4.2](#))
- Compare mode (see [Section 20.4.3](#))
- Compare output path (see [Section 20.4.4](#))
- Shadow register transfer (see [Section 20.4.5](#))
- T13 counter register description (see [Section 20.4.6](#))

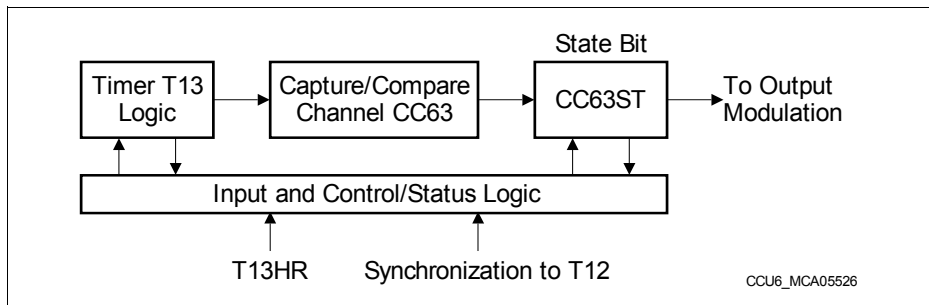


Figure 20-23 Overview Diagram of the Timer T13 Block

20.4.1 T13 Overview

Figure 20-24 shows a detailed block diagram of Timer T13. The functions of the timer T12 block are controlled by bits in registers **TCTR0L**, **TCTR0H**, **TCTR2L**, **TCTR2H**, **TCTR4L**, **TCTR4H**, and **PISEL2**.

Timer T13 receives its input clock, f_{T13} , from the module clock f_{CC6} via a programmable prescaler and an optional 1/256 divider or from an input signal T13HR. T13 can only count up (similar to the Edge-Aligned mode of T12).

Via a comparator, the timer T13 Counter Register **T13L**, **T13H** is connected to the Period Register **T13PRL**, **T13PRH**. This register determines the maximum count value for T13. When T13 reaches the period value, signal T13_PM (T13 Period Match) is generated and T13 is cleared to 0000_H with the next T13 clock edge. The Period Register receives a new period value from its Shadow Period Register, T13PS, that is loaded via software. The transfer of a new period value from the shadow register into T13PR is controlled via the 'T13 Shadow Transfer' control signal, T13_ST. The generation of this signal depends

Capture/Compare Unit 6 (CCU6)

on the associated control bit STE13. Providing a shadow register for the period value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters (refer to [Table 20.4.5](#)).

Another signal indicates whether the counter contents are equal to 0000_H (T13_ZM).

A Single-Shot control bit, T13SSC, enables an automatic stop of the timer when the current counting period is finished (see [Figure 20-24](#)).

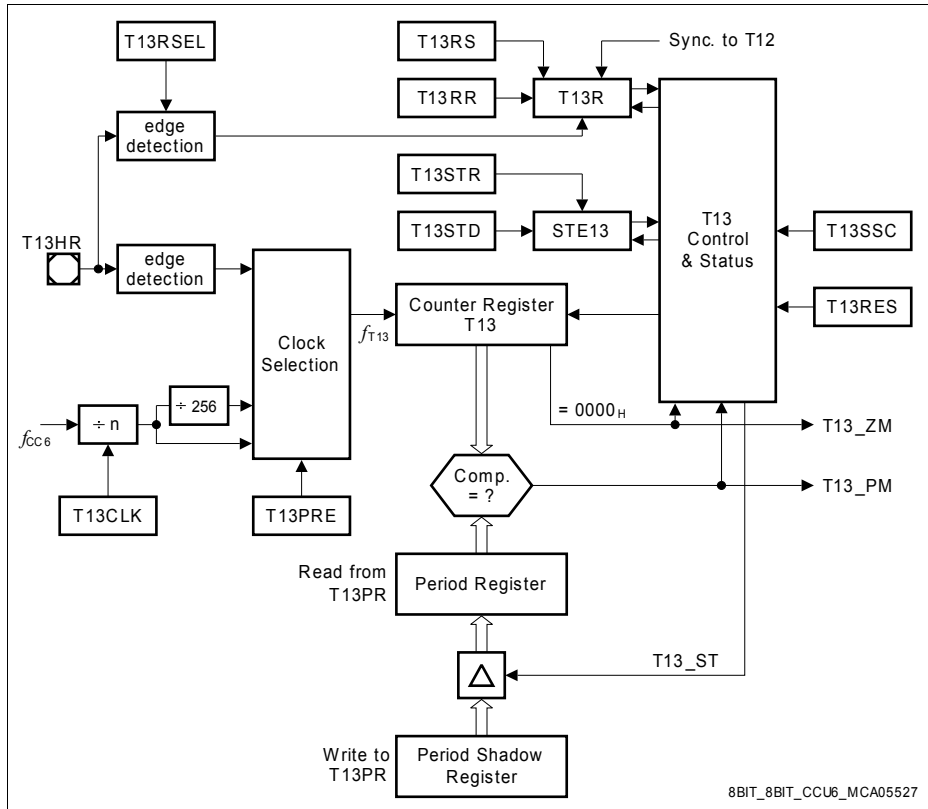


Figure 20-24 T13 Counter Logic and Period Comparators

The start or stop of T13 is controlled by the Run bit, T13R. This control bit can be set by software via the associated set/clear bits T13RS or T13RR in register [TCTR4L](#), [TCTR4H](#), or it is cleared by hardware according to preselected conditions (single-shot mode).

The timer T13 run bit T13R must not be set while the applied T13 period value is zero. Bit T13R can be set automatically if an event of T12 is detected to synchronize T13

Capture/Compare Unit 6 (CCU6)

timings to T12 events, e.g. to generate a programmable delay via T13 after an edge of a T12 compare channel before triggering an AD conversion (T13 can trigger ADC conversions).

Timer T13 can be cleared to 0000_H via control bit T13RES. Setting this write-only bit only clears the timer contents, but has no further effects, e.g., it does not stop the timer.

The generation of the T13 shadow transfer control signal, T13_ST, is enabled via bit STE13. This bit can be set or cleared by software indirectly through its associated set/reset control bits T13STR and T13STD.

Two bit fields, T13TEC and T13TED, control the synchronization of T13 to Timer T12 events. T13TEC selects the trigger event, while T13TED determines for which T12 count direction the trigger should be active.

While Timer T13 is running, write accesses to the count register T13 are not taken into account. If T13 is stopped, write actions to register T13 are immediately taken into account.

Note: The T13 Period Register and its associated shadow register are located at the same physical address. A write access to this address targets the Shadow Register, while a read access reads from the actual period register.

20.4.2 T13 Counting Scheme

This section describes the clocking and the counting capabilities of T13.

20.4.2.1 T13 Counting

The period of the timer is determined by the value in the period Register T13PR according to the following formula:

$$T13_{PER} = \text{<Period-Value>} + 1; \text{ in } T13 \text{ clocks } (f_{T13}) \quad (20.3)$$

Timer T13 can only count up, comparable to the Edge-Aligned mode of T12. This leads to very simple 'counting rule' for the T13 counter:

- The counter is cleared with the next T13 clock edge if a Period-Match is detected. The counting direction is always upwards.

The behavior of T13 is illustrated in [Figure 20-25](#).

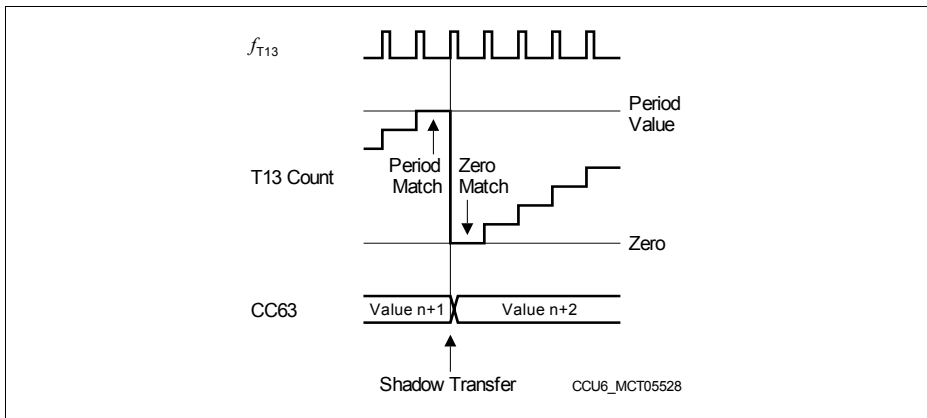


Figure 20-25 T13 Counting Sequence

20.4.2.2 Single-Shot Mode

In Single-Shot Mode, the timer run bit T13R is cleared by hardware. If bit T13SSC = 1, the timer T13 will stop when the current timer period is finished.

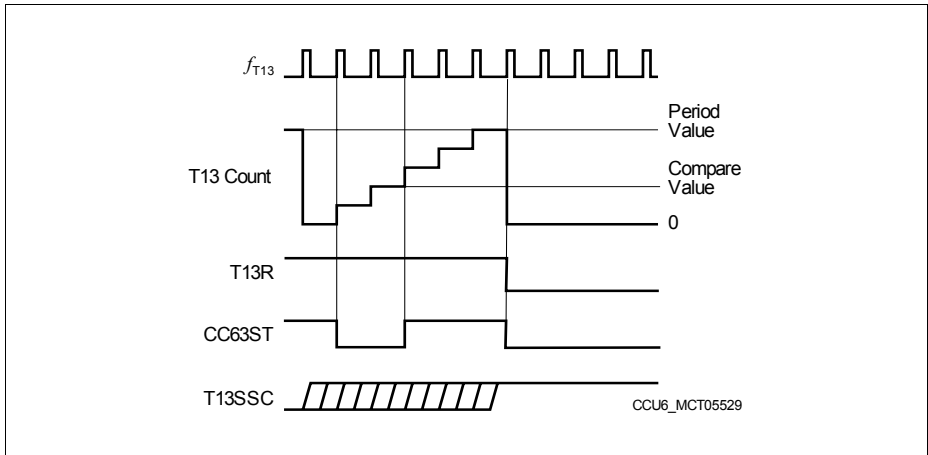


Figure 20-26 Single-Shot Operation of Timer T13

20.4.2.3 Synchronization to T12

Timer T13 can be synchronized to a T12 event. Bit fields T13TEC and T13TED select the event that is used to start Timer T13. The selected event sets bit T13R via HW, and T13 starts counting. Combined with the Single-Shot mode, this feature can be used to generate a programmable delay after a T12 event.

Figure 20-27 shows an example for the synchronization of T13 to a T12 event. Here, the selected event is a compare-match (compare value = 2) while counting up. The clocks of T12 and T13 can be different (other prescaler factor); the figure shows an example in which T13 is clocked with half the frequency of T12.

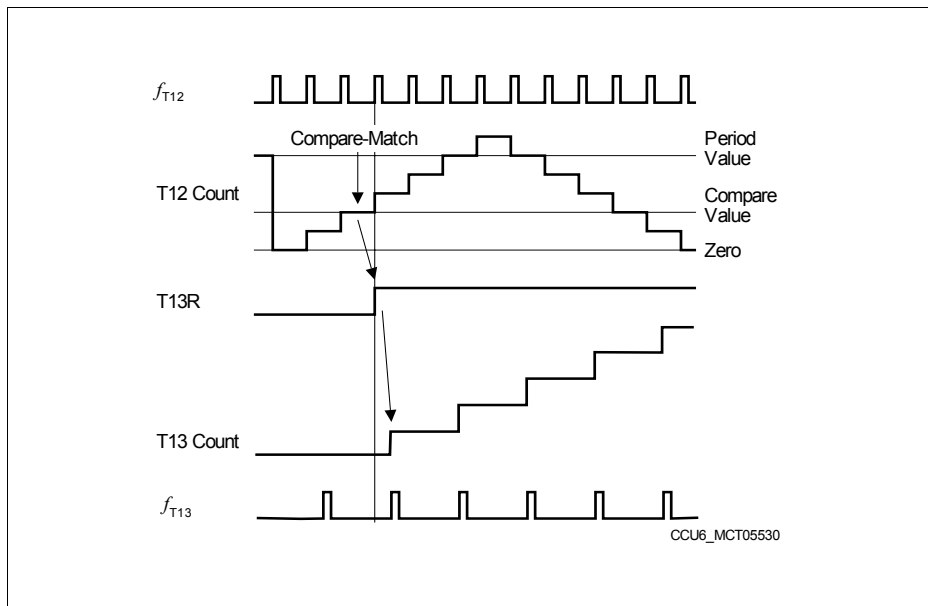


Figure 20-27 Synchronization of T13 to T12 Compare Match

Bit field T13TEC selects the trigger event to start T13 (automatic set of T13R for synchronization to T12 compare signals) according to the combinations shown in [Table 20-11](#). Bit field T13TED additionally specifies for which count direction of T12 the selected trigger event should be regarded (see [Table 20-12](#)).

Capture/Compare Unit 6 (CCU6)

Table 20-11 T12 Trigger Event Selection

T13TEC	Selected Event
000 _B	None
001 _B	T12 Compare Event on Channel 0 (CM_CC60)
010 _B	T12 Compare Event on Channel 1 (CM_CC61)
011 _B	T12 Compare Event on Channel 2 (CM_CC62)
100 _B	T12 Compare Event on any Channel (0, 1, 2)
101 _B	T12 Period-Match (T12_PM)
110 _B	T12 Zero-Match while counting up (T12_ZM and CDIR = 0)
111 _B	Any Hall State Change

Table 20-12 T12 Trigger Event Additional Specifier

T13TED	Selected Event Specifier
00 _B	Reserved, no action
01 _B	Selected event is active while T12 is counting up (CDIR = 0)
10 _B	Selected event is active while T12 is counting down (CDIR = 1)
11 _B	Selected event is active independently of the count direction of T12

20.4.3 T13 Compare Mode

Associated with Timer T13 is one compare channel, that can perform compare operations with regard to the contents of the T13 counter.

Figure 20-23 gives an overview on the T13 channel in Compare Mode. The channel is connected to the T13 counter register via an equal-to comparator, generating a compare match signal when the contents of the counter matches the contents of the compare register.

The channel consists of the comparator and a double register structure - the actual compare register, **CC63RL**, **CC63RH**, feeding the comparator, and an associated shadow register, **CC63SRL**, **CC63SRH**, that is preloaded by software and transferred into the compare register when signal T13 shadow transfer, T13_ST, gets active. Providing a shadow register for the compare value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters.

Associated with the channel is a State Bit, **CMPSTATL.CC63ST**, holding the status of the compare operation. **Figure 20-28** gives an overview on the logic for the State Bit.

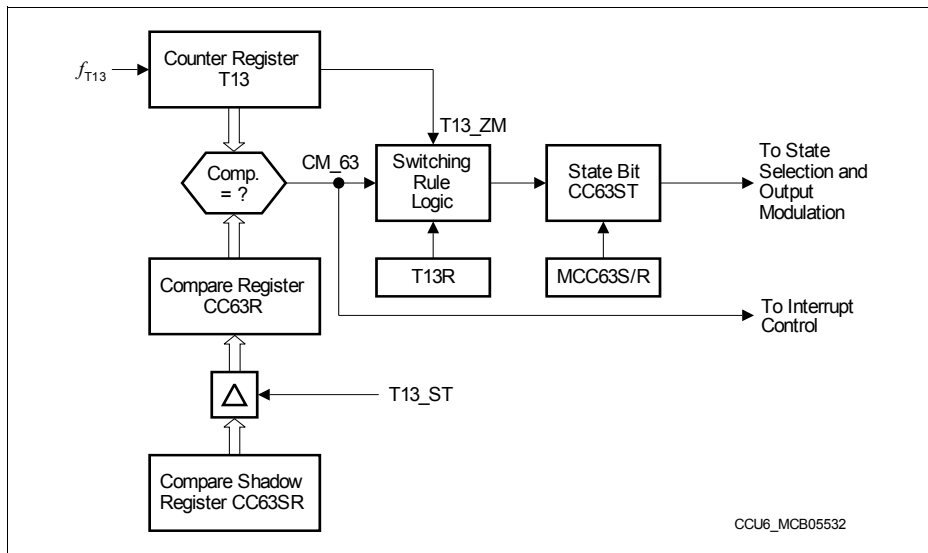


Figure 20-28 T13 State Bit Block Diagram

A compare interrupt event CM_63 is signaled when a compare match is detected. The actual setting of a State Bit has no influence on the interrupt generation.

The inputs to the switching rule logic for the CC63ST bit are the timer run bit (T13R), the timer zero-match signal (T13_ZM), and the actual individual compare-match signal

Capture/Compare Unit 6 (CCU6)

CM_63. In addition, the state bit can be set or cleared by software via bits MCC63S and MCC63R in register **CMPMODIFL**, **CMPMODIFH**.

A modification of the State Bit CC63ST by hardware is only possible while Timer T13 is running ($T13R = 1$). If this is the case, the following switching rules apply for setting and resetting the State Bit in Compare Mode:

State Bit **CC63ST** is **set** to 1

- with the next T13 clock (f_{T13}) after a compare-match (T13 is always counting up) (i.e., when the counter is incremented above the compare value);
- with the next T13 clock (f_{T13}) after a zero-match AND a parallel compare-match.

State Bit **CC63ST** is **cleared** to 0

- with the next T13 clock (f_{T13}) after a zero-match AND NO parallel compare-match.

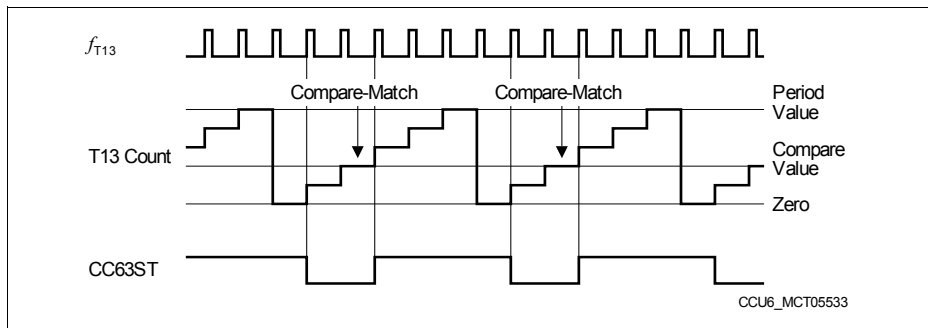


Figure 20-29 T13 Compare Operation

20.4.4 Compare Mode Output Path

Figure 20-30 gives an overview on the signal path from the channel State Bit CC63ST to its output pin COUT63. As illustrated, a user can determine the desired output behavior in relation to the current state of CC63ST. Please refer to [Section 20.3.4.3](#) for detailed information on the output modulation for T12 signals.

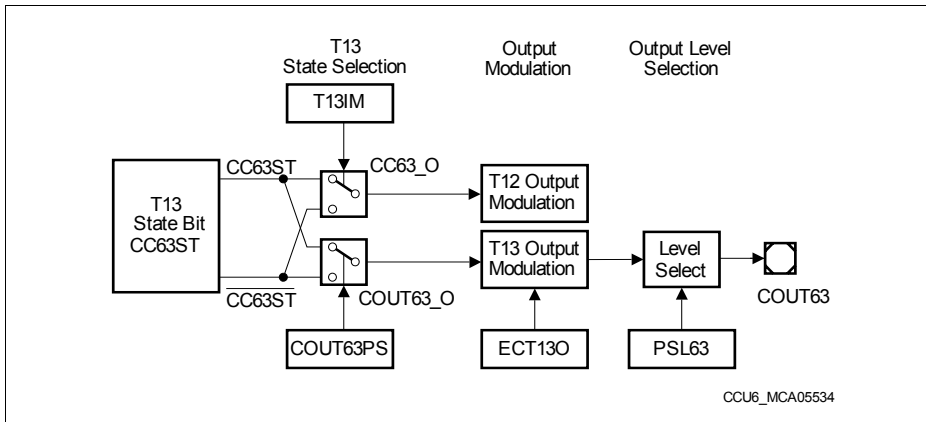


Figure 20-30 Channel 63 Output Path

The output line COUT63_O can generate a T13 PWM at the output pin COUT63. The signal CC63_O can be used to modulate the T12-related output signals with a T13 PWM. In order to decouple COUT63 from the internal modulation, the compare state leading to an active signal can be selected independently by bits T13IM and COUT63PS.

The last block of the data path is the Output Modulation block. Here, the modulation source T13 and the trap functionality are combined and control the actual level of the output pin COUT63 (see [Figure 20-31](#)):

- The **T13 related compare signal** COUT63_O delivered by the T13 state selection with the enable bit **MODCTR.H.ECT13O**
- The **trap state** TRPS with an individual enable bit **TRPCTR.H.TRPEN13**

If the modulation input signal COUT63_O is enabled (ECT13O = 1) and is at passive state, the modulated is also in passive state. If the modulation input is not enabled, the output is in passive state.

If the Trap State is active (TRPS = 1), then the output enabled for the trap signal (by TRPEN13 = 1) is set to the passive state.

The output of the modulation control block is connected to a level select block. It offers the option to determine the actual output level of a pin, depending on the state of the output line (decoupling of active/passive state and output polarity) as specified by the Passive State Select bit **PSLR.PSL63**. If the modulated output signal is in the passive

Capture/Compare Unit 6 (CCU6)

state, the level specified directly by PSL63 is output. If it is in the active state, the inverted level of PSL63 is output. This allows the user to adapt the polarity of an active output signal to the connected circuitry.

The PSL63 bit has a shadow register to allow for updates with the T13 shadow transfer signal (T13_ST) without undesired pulses on the output lines. A read action returns the actually used value, whereas a write action targets the shadow bit. Providing a shadow register for the PSL value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters.

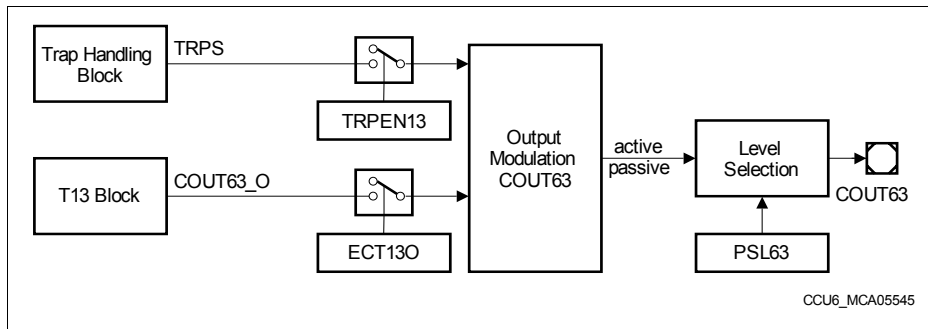


Figure 20-31 T13 Output Modulation

20.4.5 T13 Shadow Register Transfer

A special shadow transfer signal (T13_ST) can be generated to facilitate updating the period and compare values of the compare channel CC63 synchronously to the operation of T13. Providing a shadow register for values defining one PWM period facilitates a concurrent update by software for all relevant parameters. The next PWM period can run with a new set of parameters. The generation of this signal is requested by software via bit **TCTR0H.STE13** (set by writing 1 to the write-only bit **TCTR4H.T13STR**, cleared by writing 1 to the write-only bit **TCTR4H.T13STD**).

When signal T13_ST is active, a shadow register transfer is triggered with the next cycle of the T13 clock. Bit STE13 is automatically cleared with the shadow register transfer.

A T13 shadow register transfer takes place (T13_ST active):

- while timer T13 is not running (T13R = 0), or
- STE13 = 1 and a Period-Match is detected while T13R = 1

Capture/Compare Unit 6 (CCU6)

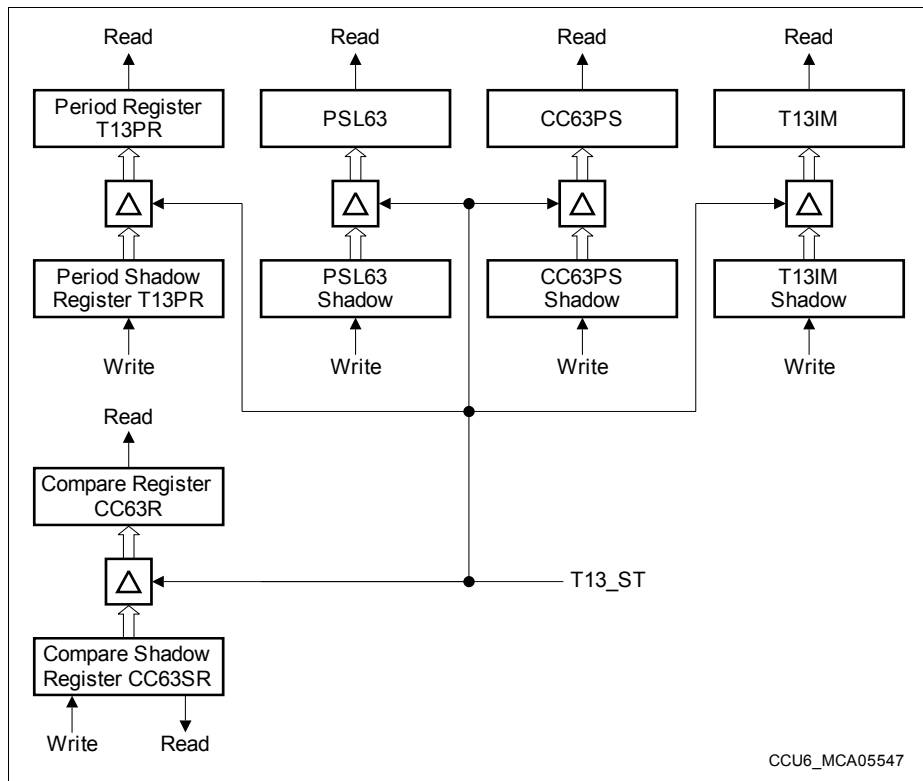


Figure 20-32 T13 Shadow Register Overview

Capture/Compare Unit 6 (CCU6)

20.4.6 T13 related Registers

20.4.6.1 T13 Counter Register

The generation of the patterns for a single channel pulse width modulation (PWM) is based on timer T13. The registers related to timer T13 can be concurrently updated (with well-defined conditions) in order to ensure consistency of the PWM signal. T13 can be synchronized to several timer T12 events.

Timer T13 only supports compare mode on its compare channel CC63.

Register T13 represents the counting value of timer T13. It can only be written while the timer T13 is stopped. Write actions while T13 is running are not taken into account. Register T13 can always be read by SW.

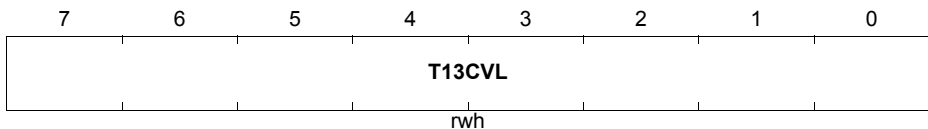
Timer T13 only supports edge-aligned mode (counting up).

T13L

Timer T13 Counter Register Low (FC_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3



Field	Bits	Type	Description
T13CVL	[7:0]	rwh	Timer 13 Counter Value This register represents the lower 8-bits of 16-bit counter value of Timer13.

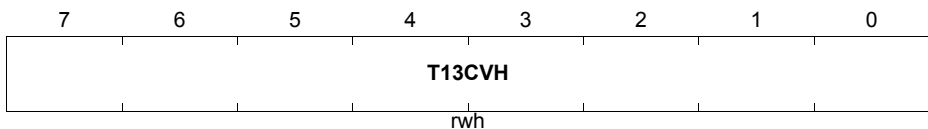
Note: While timer T13 is stopped, the internal clock divider is reset in order to ensure reproducible timings and delays.

T13H

Timer T13 Counter Register High (FD_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3



Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
T13CVH	[7:0]	rwh	Timer 13 Counter Value This register represents the upper 8-bits of 16-bit counter value of Timer13.

Note: While timer T13 is stopped, the internal clock divider is reset in order to ensure reproducible timings and delays.

Capture/Compare Unit 6 (CCU6)
20.4.6.2 Period Register

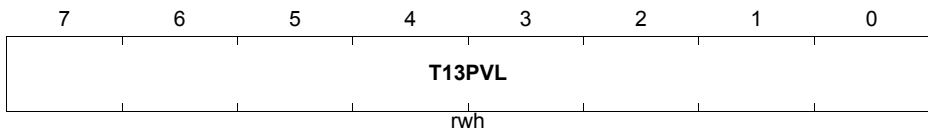
The generation of the patterns for a single channel pulse width modulation (PWM) is based on timer T13. The registers related to timer T13 can be concurrently updated (with well-defined conditions) in order to ensure consistency of the PWM signal. T13 can be synchronized to several timer T12 events.

Timer T13 only supports compare mode on its compare channel CC63.

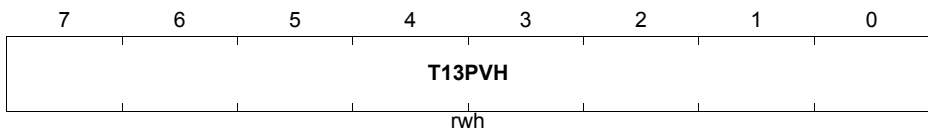
Register T13 represents the counting value of timer T13. It can only be written while the timer T13 is stopped. Write actions while T13 is running are not taken into account.

Register T13 can always be read by SW.

Timer T13 only supports edge-aligned mode (counting up).

T13PRL
Timer T13 Period Register Low
(9E_H)
Reset Value: 00_H
RMAP: 0, PAGE: 1


Field	Bits	Type	Description
T13PVL	[7:0]	rwh	T13 Period Value The value T13PV defines the lower 8-bits of counter value for T13 leading to a period-match. When reaching this value, the timer T13 is set to zero.

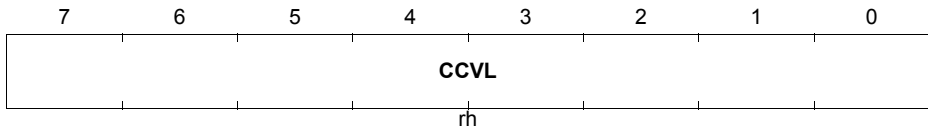
T13PRH
Timer T13 Period Register High
(9F_H)
Reset Value: 00_H
RMAP: 0, PAGE: 1


Capture/Compare Unit 6 (CCU6)

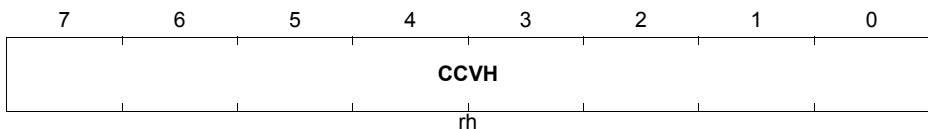
Field	Bits	Type	Description
T13PVH	[7:0]	rwh	T13 Period Value The value T13PV defines the upper 8-bits of counter value for T13 leading to a period-match. When reaching this value, the timer T13 is set to zero.

Capture/Compare Unit 6 (CCU6)
20.4.6.3 Compare Register

Registers CC63RL/H is the actual compare register for T13. The values stored in CC63RL/H is compared to the counter value of T13. The State Bit CC63ST is located in register **CMPSTATL**.

CC63RL
Compare Register for T13 Low
(9A_H)
Reset Value: 00_H
RMAP: 0, PAGE: 1


Field	Bits	Type	Description
CCVL	[7:0]	rh	Channel CC63 Compare Value The bit field CCV contains the lower 8-bits value, that is compared to the T13 counter value.

CC63RH
Compare Register for T13 High
(9B_H)
Reset Value: 00_H
RMAP: 0, PAGE: 1


Field	Bits	Type	Description
CCVH	[7:0]	rh	Channel CC63 Compare Value The bit field CCV contains the upper 8-bits value, that is compared to the T13 counter value.

20.4.6.4 Compare Shadow Register

The register CC63RL/H can only be read by SW, the modification of the value is done by a shadow register transfer from register CC63SRL/H. The corresponding shadow register CC63SRL/H can be read and written by SW.

Capture/Compare Unit 6 (CCU6)

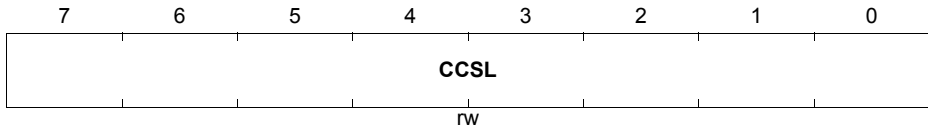
CC63SRL

Compare Shadow Register for T13 Low

(9A_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0



Field	Bits	Type	Description
CCSL	[7:0]	rw	Shadow Register for Channel CC63 Compare Value The bit field contents of CCSL is transferred to the lower 8-bits of bit field CCV during a shadow transfer.

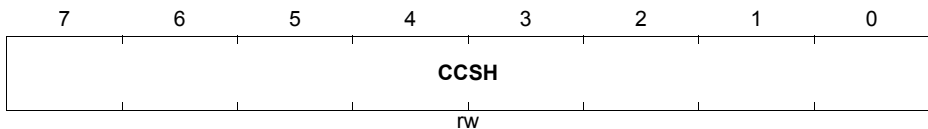
CC63SRH

Compare Shadow Register for T13 High

(9B_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0



Field	Bits	Type	Description
CCSH	[7:0]	rw	Shadow Register for Channel CC63 Compare Value The bit field contents of CCSH is transferred to the upper 8-bits of bit field CCV during a shadow transfer.

20.5 Trap Handling

The trap functionality permits the PWM outputs to react on the state of the input signal $\overline{\text{CTRAP}}$. This functionality can be used to switch off the power devices if the trap input becomes active (e.g. to perform an emergency stop). The trap handling and the effect on the output modulation are controlled by the bits in the trap control register **TRPCTRL**, **TRPCTRH**. The trap flags TRPF and TRPS are located in register **ISH** and can be set/cleared by SW by writing to registers **ISSH** and **ISRH**.

Figure 20-33 gives an overview on the trap function.

The Trap Flag TRPF monitors the trap input and initiates the entry into the Trap State. The Trap State Bit TRPS determines the effect on the outputs and controls the exit of the Trap State.

When a trap condition is detected ($\overline{\text{CTRAP}} = 0$) and the input is enabled (TRPPEN = 1), both, the Trap Flag TRPF and the Trap State Bit TRPS, are set to 1 (trap state active). The output of the Trap State Bit TRPS leads to the Output Modulation Blocks (for T12 and for T13) and can there deactivate the outputs (set them to the passive state). Individual enable control bits for each of the six T12-related outputs and the T13-related output facilitate a flexible adaptation to the application needs.

There are a number of different ways to exit the Trap State. This offers SW the option to select the best operation for the application. Exiting the Trap State can be done either immediately when the trap condition is removed ($\overline{\text{CTRAP}} = 1$ or TRPPEN = 0), or under software control, or synchronously to the PWM generated by either Timer T12 or Timer T13.

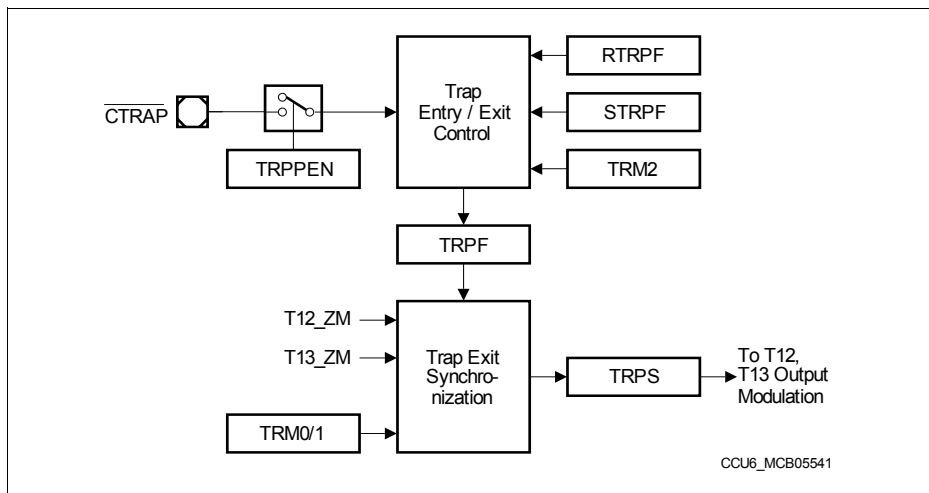


Figure 20-33 Trap Logic Block Diagram

Capture/Compare Unit 6 (CCU6)

Clearing of TRPF is controlled by the mode control bit TRPM2. If $TRPM2 = 0$, TRPF is automatically cleared by HW when CTRAP returns to the inactive level (CTRAP = 1) or if the trap input is disabled (TRPPEN = 0). When $TRPM2 = 1$, TRPF must be reset by SW after CTRAP has become inactive.

Clearing of TRPS is controlled by the mode control bits TRPM1 and TRPM0 (located in the Trap Control Register TRPCTRL/H). A reset of TRPS terminates the Trap State and returns to normal operation. There are three options selected by TRPM1 and TRPM0. One is that the Trap State is left immediately when the Trap Flag TRPF is cleared, without any synchronization to timers T12 or T13. The other two options facilitate the synchronization of the termination of the Trap State to the count periods of either Timer T12 or Timer T13. **Figure 20-34** gives an overview on the associated operation.

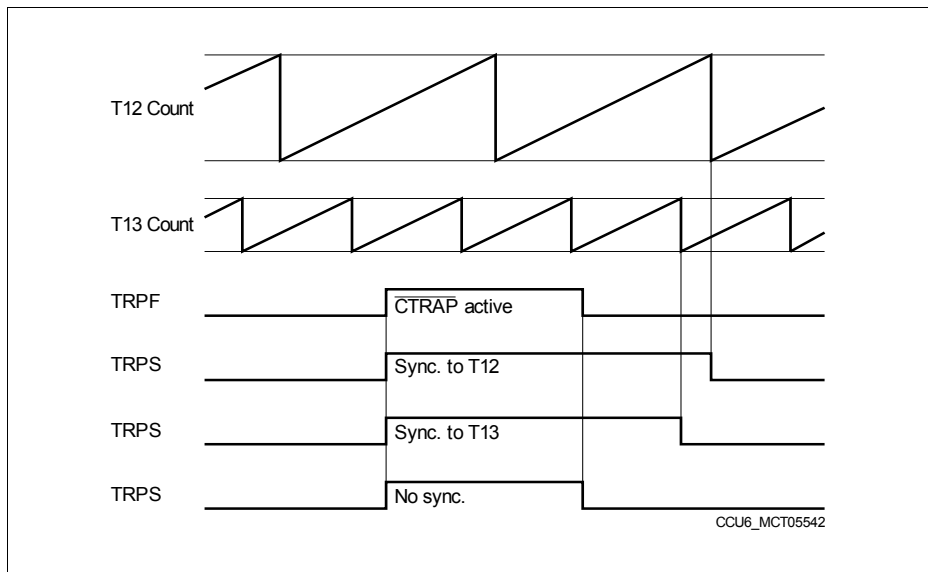


Figure 20-34 Trap State Synchronization (with TRM2 = 0)

20.6 Multi-Channel Mode

The Multi-Channel mode offers the possibility to modulate all six T12-related output signals with one instruction. The bits in bit field **MCOUTL.MCMP** are used to specify the outputs that may become active. If Multi-Channel mode is enabled (bit **MODCTRL.MCMEN** = 1), only those outputs may become active, that have a 1 at the corresponding bit position in bit field **MCMP**.

This bit field has its own shadow bit field **MCOUTSL.MCMPS**, that can be written by software. The transfer of the new value in **MCMP** to the bit field **MCMP** can be triggered by, and synchronized to, T12 or T13 events. This structure permits the software to write the new value, that is then taken into account by the hardware at a well-defined moment and synchronized to a PWM signal. This avoids unintended pulses due to unsynchronized modulation sources.

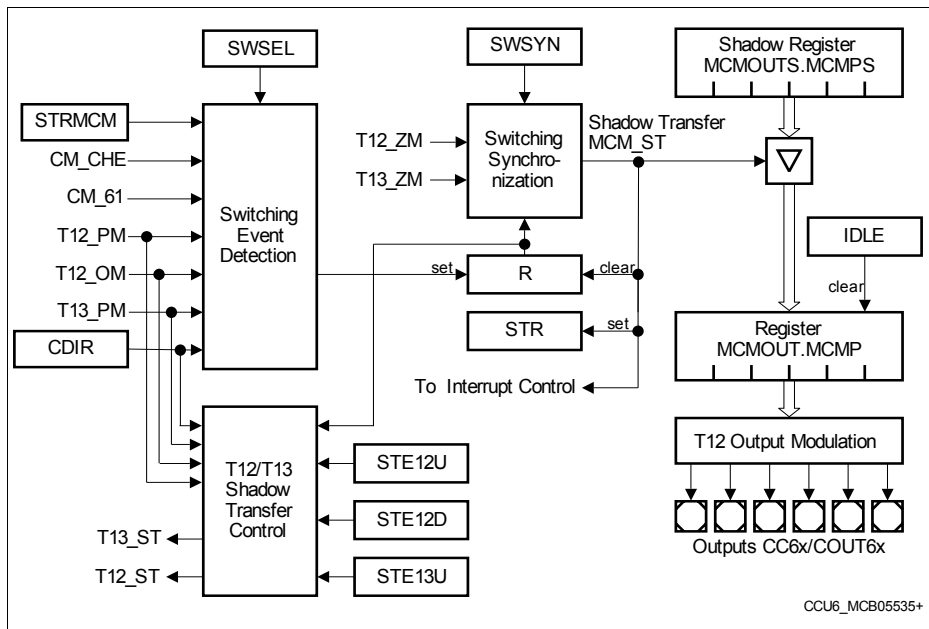


Figure 20-35 Multi-Channel Mode Block Diagram

Figure 20-35 shows the functional blocks for the Multi-Channel operation, controlled by bit fields in register **MCMCTR**. The event that triggers the update of bit field **MCMP** is chosen by **SWSEL**. In order to synchronize the update of **MCMP** to a PWM generated by T12 or T13, bit field **SWSYN** allows the selection of the synchronization event leading to the transfer from **MCMPS** to **MCMP**. Due to this structure, an update takes place with a new PWM period. A remainder flag **R** is set when the selected switching event occurs

Capture/Compare Unit 6 (CCU6)

(the event is not necessarily synchronous to the modulating PWM), and is cleared when the transfer takes place. This flag can be monitored by software to check for the status of this logic block. If the shadow transfer from MCMPS to MCMP takes place, bit **ISH.STR** becomes set and an interrupt can be generated.

In addition to the Multi-Channel shadow transfer event MCM_ST, the shadow transfers for T12 (T12_ST) and T13 (T13_ST) can be generated to allow concurrent updates of applied duty cycles for T12 and/or T13 modulation and Multi-Channel patterns.

If it is explicitly desired, the update takes place immediately with the occurrence of the selected event when the direct synchronization mode is selected. The update can also be requested by software by writing to bit field MCMPS with the shadow transfer request bit STRMCM = 1. The option to trigger an update by SW is possible for all settings of SWSEL.

By using the direct mode and bit STRMCM = 1, the update takes place completely under software control.

The event selection and synchronization options are summarized in [Table 20-13](#) and [Table 20-14](#).

Table 20-13 Multi-Channel Mode Switching Event Selection

SWSEL	Selected Event (see register MCMCTR)
000 _B	No automatic event detection
001 _B	Correct Hall Event (CM_CHE) detected at input signals CCPOSx without additional delay
010 _B	T13 Period-Match (T13_PM)
011 _B	T12 One-Match while counting down (T12_OM and CDIR = 1)
100 _B	T12 Compare Channel 1 Event while counting up (CM_61 and CDIR = 0) to support the phase delay function by CC61 for block commutation mode.
101 _B	T12 Period-Match while counting up (T12_PM and CDIR = 0)
110 _B , 111 _B	Reserved, no action

Table 20-14 Multi-Channel Mode Switching Synchronization

SWSYN	Synchronization Event (see register MCMCTR)
00 _B	Direct Mode: the trigger event directly causes the shadow transfer
01 _B	T13 Zero-Match (T13_ZM), the MCM shadow transfer is synchronized to a T13 PWM

Capture/Compare Unit 6 (CCU6)

Table 20-14 Multi-Channel Mode Switching Synchronization (cont'd)

SWSYN	Synchronization Event (see register MCMCTR)
10 _B	T12 Zero-Match (T12_ZM), the MCM shadow transfer is synchronized to a T12 PWM
11 _B	Reserved, no action

20.7 Hall Sensor Mode

For Brushless DC-Motors in block commutation mode, the Multi-Channel Mode has been introduced to provide efficient means for switching pattern generation. These patterns need to be output in relation to the angular position of the motor. For this, usually Hall sensors or Back-EMF sensing are used to determine the angular rotor position. The CCU6 provides three inputs, CCPOS0, CCPOS1, and CCPOS2, that can be used as inputs for the Hall sensors or the Back-EMF detection signals.

There is a strong correlation between the motor position and the output modulation pattern. When a certain position of the motor has been reached, indicated by the sampled Hall sensor inputs (the Hall pattern), the next, pre-determined Multi-Channel Modulation pattern has to be output. Because of different machine types, the modulation pattern for driving the motor can vary. Therefore, it is wishful to have a wide flexibility in defining the correlation between the Hall pattern and the corresponding Modulation pattern. Furthermore, a hardware mechanism significantly reduces the CPU for block-commutation.

The CCU6 offers the flexibility by having a register containing the currently assumed Hall pattern (CURH), the next expected Hall pattern (EXPH) and the corresponding output pattern (MCMP). A new Modulation pattern is output when the sampled Hall inputs match the expected ones (EXPH). To detect the next rotation phase (segment for block commutation), the CCU6 monitors the Hall inputs for changes. When the next expected Hall pattern is detected, the next corresponding Modulation pattern is output.

To increase for noise immunity (to a certain extend), the CCU6 offers the possibility to introduce a sampling delay for the Hall inputs. Some changes of the Hall inputs are not leading to the expected Hall pattern, because they are only short spikes due to noise. The Hall pattern compare logic compares the Hall inputs to the next expected pattern and also to the currently assumed pattern to filter out spikes.

For the Hall and Modulation output patterns, a double-register structure is implemented. While register **MCMOUTL**, **MCMOUTH** holds the actually used values, its shadow register **MCMOUTSL**, **MCMOUTSH** can be loaded by software from a pre-defined table, holding the appropriate Hall and Modulation patterns for the given motor control.

A transfer from the shadow register into register MCMOUT can take place when a correct Hall pattern change is detected. Software can then load the next values into register MCMOUTS. It is also possible by software to force a transfer from MCMOUTS into MCMOUT.

Note: The Hall input signals CCPOSx and the CURH and EXPH bit fields are arranged in the following order:

CCPOS0 corresponds to CURH.0 (LSB) and EXPH.0 (LSB)

CCPOS1 corresponds to CURH.1 and EXPH.1

CCPOS2 corresponds to CURH.2 (MSB) and EXPH.2 (MSB)

Capture/Compare Unit 6 (CCU6)

20.7.1 Hall Pattern Evaluation

The Hall sensor inputs CCPOSx can be permanently monitored via an edge detection block (with the module clock f_{CC6}). In order to suppress spikes on the Hall inputs due to noise in rugged inverter environment, two optional noise filtering methods are supported by the Hall logic (both methods can be combined).

- Noise filtering with delay:
For this function, the mode control bit fields MSEL6x for all T12 compare channels must be programmed to 1000_B and DBYP = 0. The selected event triggers Dead-Time Counter 0 to generate a programmable delay (defined by bit field DTM). When the delay has elapsed, the evaluation signal HCRDY becomes activated.
Output modulation with T12 PWM signals is not possible in this mode.
- Noise filtering by synchronization to PWM:
The Hall inputs are not permanently monitored by the edge detection block, but samples are taken only at defined points in time during a PWM period. This can be used to sample the Hall inputs when the switching noise (due to PWM) does not disturb the Hall input signals.

If neither the delay function of Dead-Time Counter 0 is not used for the Hall pattern evaluation nor the Hall mode for Brushless DC-Drive control is enabled, the timer T12 block is available for PWM generation and output modulation.

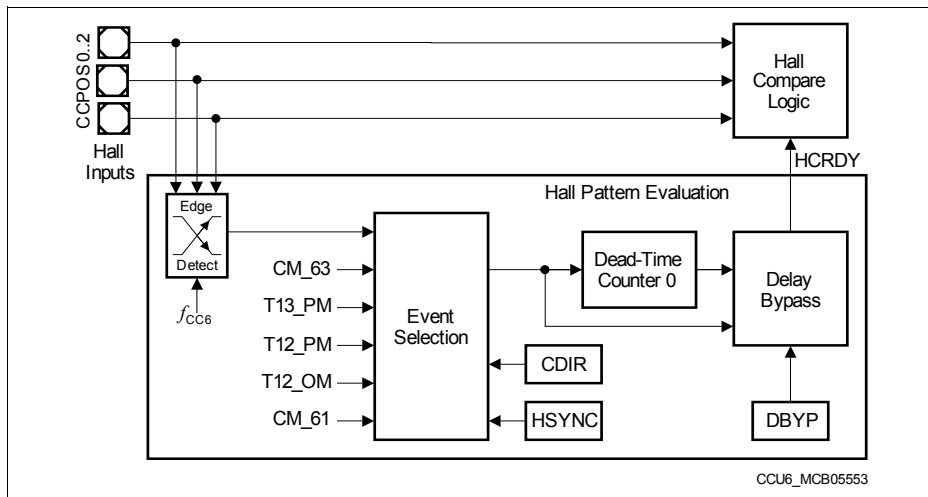


Figure 20-36 Hall Pattern Evaluation

If the evaluation signal HCRDY (Hall Compare Ready, see [Figure 20-37](#)) becomes activated, the Hall inputs are sampled and the Hall compare logic starts the evaluation of the Hall inputs.

Capture/Compare Unit 6 (CCU6)

Figure 20-36 illustrates the events for Hall pattern evaluation and the noise filter logic, **Table 20-15** summarizes the selectable trigger input signals.

Table 20-15 Hall Sensor Mode Trigger Event Selection

HSYNC	Selected Event (see register T12MSELL , T12MSELH)
000 _B	Any edge at any of the inputs CCPOSx, independent from any PWM signal (permanent check).
001 _B	A T13 Compare-Match (CM_63).
010 _B	A T13 Period-Match (T13_PM).
011 _B	Hall sampling triggered by HW sources is switched off.
100 _B	A T12 Period-Match while counting up (T12_PM and CDIR = 0).
101 _B	A T12 One-Match while counting down (T12_OM and CDIR = 1).
110 _B	A T12 Compare-Match of compare channel CC61 while counting up (CM_61 and CDIR = 0).
111 _B	A T12 Compare-Match of compare channel CC61 while counting down (CM_61 and CDIR = 1).

Capture/Compare Unit 6 (CCU6)

20.7.2 Hall Pattern Compare Logic

Figure 20-37 gives an overview on the double-register structure and the pattern compare logic. Software writes the next modulation pattern (MCMPS) and the corresponding current (CURHS) and expected (EXPHS) Hall patterns into the shadow register MCMOUTS. Register MCMOUT holds the actually used values CURH and EXPH. The modulation pattern MCMP is provided to the T12 Output Modulation block. The current (CURH) and expected (EXPH) Hall patterns are compared to the sampled Hall sensor inputs (visible in register [CMPSTATL](#), [CMPSTATH](#)). Sampling of the inputs and the evaluation of the comparator outputs is triggered by the evaluation signal HCRDY (Hall Compare Ready), that is detailed in the next section.

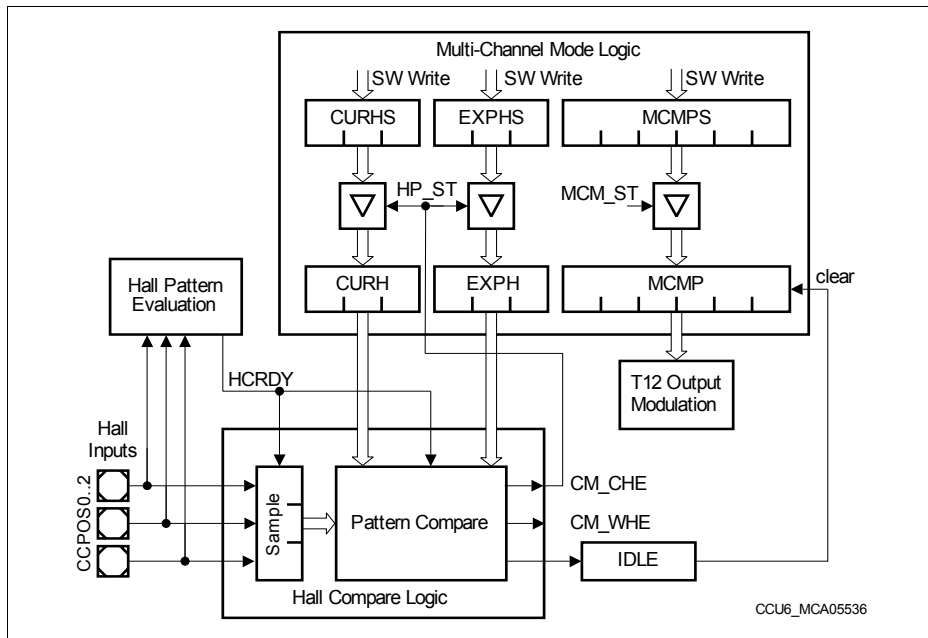


Figure 20-37 Hall Pattern Compare Logic

- If the sampled Hall pattern matches the value programmed in CURH, the detected transition was a spike (no Hall event) and no further actions are necessary.
- If the sampled Hall pattern matches the value programmed in EXPH, the detected transition was the expected event (correct Hall event CM_CHE) and the MCMP value has to change.
- If the sampled Hall pattern matches neither CURH nor EXPH, the transition was due to a major error (wrong Hall event CM_CWE) and can lead to an emergency shut down (IDLE).

Capture/Compare Unit 6 (CCU6)

At every correct Hall event (CM_CHE), the next Hall patterns are transferred from the shadow register MCMOUTS into MCMOUT (Hall pattern shadow transfer HP_ST), and a new Hall pattern with its corresponding output pattern can be loaded (e.g. from a predefined table in memory) by software into MCMOUTS. For the Modulation patterns, signal MCM_ST is used to trigger the transfer.

Loading this shadow register can also be done by writing MCMOUTS.STRHP = 1 (for EXPH and CURH) or MCMOUTS.STRMCMP = 1 (for MCMP).

20.7.3 Hall Mode Flags

Depending on the Hall pattern compare operation, a number of flags are set in order to indicate the status of the module and to trigger further actions and interrupt requests.

Flag **ISH.CHE** (Correct Hall Event) is set by signal CM_CHE when the sampled Hall pattern matches the expected one (EXPH). This flag can also be set by SW by setting bit **ISSH.SCHE** = 1. If enabled by bit **INENH.ENCHE** = 1, the set signal for CHE can also generate an interrupt request to the CPU. Bit field **INPL.INPCHE** defines which service request output becomes activated in case of an interrupt request. To clear flag CHE, SW needs to write **ISRH.RCHE** = 1.

Flag **IS.WHE** indicates a Wrong Hall Event. Its handling for flag setting and resetting as well as interrupt request generation are similar to the mechanism for flag CHE.

The implementation of flag STR is done in the same way as for CHE and WHE. This flag is set by HW by the shadow transfer signal MCM_ST (see also [Figure 20-35](#)).

Please note that for flags CHE, WHE, and STR, the interrupt request generation is triggered by the set signal for the flag. That means, a request can be generated even if the flag is already set. There is no need to clear the flag in order to enable further interrupt requests.

The implementation for the IDLE flag is different. It is set by HW through signal CM_WHE if enabled by bit ENIDLE. Software can also set the flag via bit SIDLE. As long as bit IDLE is set, the modulation pattern field MCMP is cleared to force the outputs to the passive state. Flag IDLE must be cleared by software by writing RIDLE = 1 in order to return to normal operation. To fully restart from IDLE mode, the transfer requests for the bit fields in register MCMOUTS to register MCMOUT have to be initiated by software via bits STRMCM and STRHP in register MCMOUTS. In this way, the release from IDLE mode is under software control, but can be performed synchronously to the PWM signal.

Capture/Compare Unit 6 (CCU6)

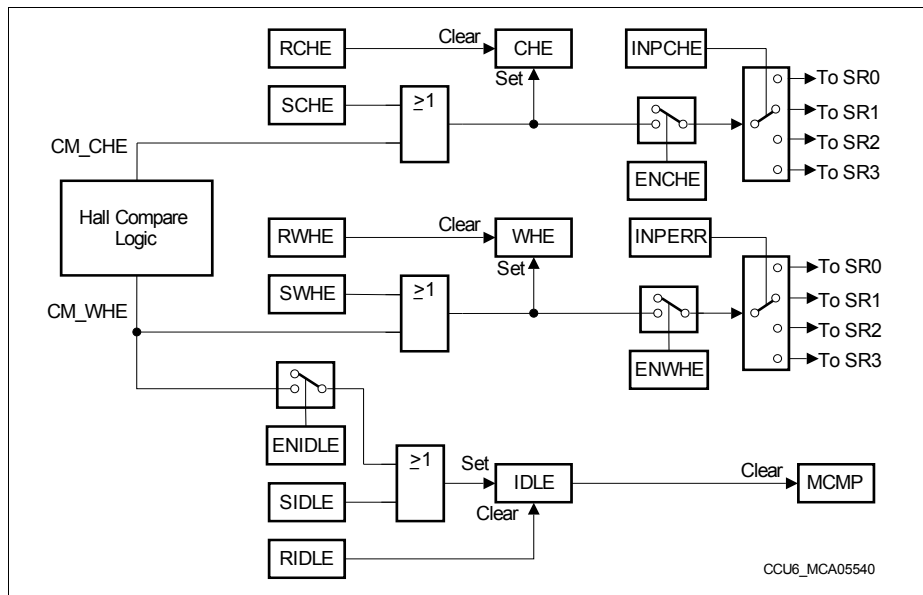


Figure 20-38 Hall Mode Flags

Capture/Compare Unit 6 (CCU6)

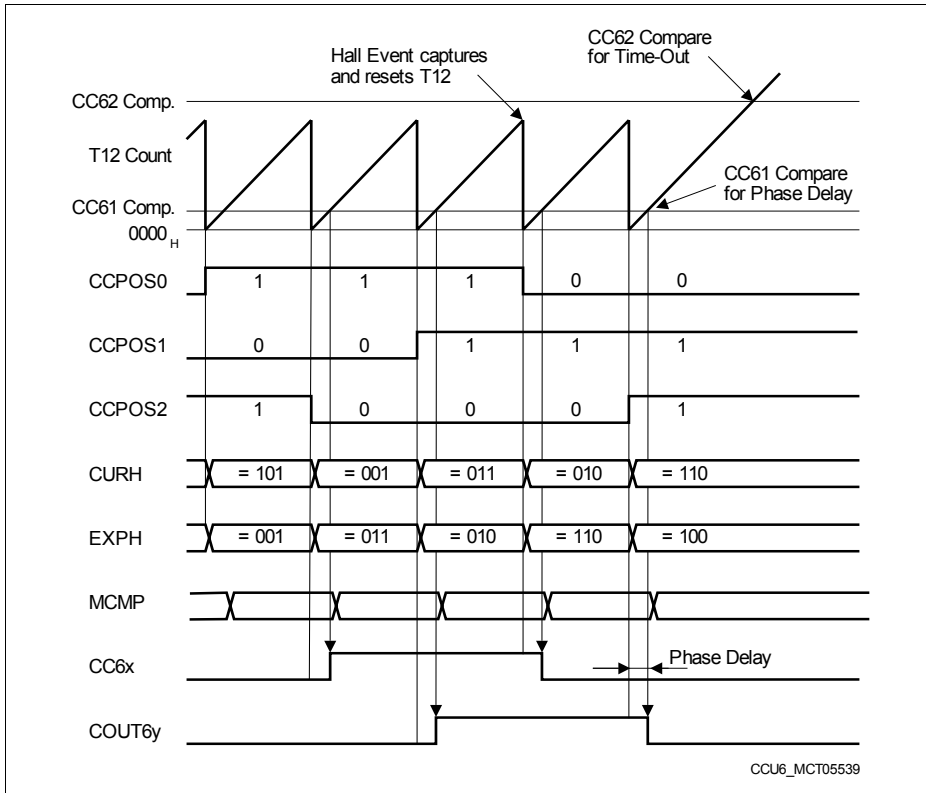


Figure 20-40 Brushless DC-Motor Control Example (all MSEL6x = 1000_B)

After the detection of an expected Hall pattern (CM_CHE active), the T12 count value is captured into channel CC60 (representing the actual rotor speed by measuring the elapsed time between the last two correct Hall events), and T12 is reset. When the timer reaches the compare value in channel CC61, the next multi-channel state is switched by triggering the shadow transfer of bit field MCMP. This trigger event can be combined with the synchronization of the next multi-channel state to the PWM source (to avoid spikes on the output lines, see [Section 20.6](#)). This compare function of channel CC61 can be used as a phase delay from the position sensor input signals to the switching of the output signals, that is necessary if a sensorless back-EMF technique or Hall sensors are used. The compare value in channel CC62 can be used as a time-out trigger (interrupt), indicating that the actual motor speed is far below the desired destination value. An abnormal load change can be detected with this feature and PWM generation can be disabled.

20.8 Modulation Control Registers

20.8.1 Modulation Control

This register contains bits enabling the modulation of the corresponding output signal by PWM pattern generated by the timers T12 and T13. Furthermore, the multi-channel mode can be enabled as additional modulation source for the output signals.

MODCTRL

Modulation Control Register Low (FC_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
MCMEN	0	T12MODEN					
rw	r	rw					

Field	Bits	Type	Description
T12MODEN	[5:0]	rw	T12 Modulation Enable These bits enable the modulation of the corresponding output signal by a PWM pattern generated by timer T12. T12MODEN0 = MODCTR.0 for output CC60 T12MODEN1 = MODCTR.1 for output COUT60 T12MODEN2 = MODCTR.2 for output CC61 T12MODEN3 = MODCTR.3 for output COUT61 T12MODEN4 = MODCTR.4 for output CC62 T12MODEN5 = MODCTR.5 for output COUT62 0 _B The modulation of the corresponding output signal by a T12 PWM pattern is disabled. 1 _B The modulation of the corresponding output signal by a T12 PWM pattern is enabled.
MCMEN	7	rw	Multi-Channel Mode Enable 0 _B The modulation of the corresponding output signal by a multi-channel pattern according to bit field MCMOUT is disabled. 1 _B The modulation of the corresponding output signal by a multi-channel pattern according to bit field MCMOUT is enabled.
0	6	r	reserved; returns 0 if read; should be written with 0;

Capture/Compare Unit 6 (CCU6)

MODCTRH

Modulation Control Register High (FD_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
ECT13O	0	T13MODEN					
rw	r	rw					

Field	Bits	Type	Description
T13MODEN	[5:0]	rw	T13 Modulation Enable These bits enable the modulation of the corresponding output signal by the PWM pattern CC63_O generated by timer T13. T13MODEN0 = MODCTR.8 for output CC60 T13MODEN1 = MODCTR.9 for output COUT60 T13MODEN2 = MODCTR.10 for output CC61 T13MODEN3 = MODCTR.11 for output COUT61 T13MODEN4 = MODCTR.12 for output CC62 T13MODEN5 = MODCTR.13 for output COUT62 0 _B The modulation of the corresponding output signal by a T13 PWM pattern is disabled. 1 _B The modulation of the corresponding output signal by a T13 PWM pattern is enabled.
ECT13O	7	rw	Enable Compare Timer T13 Output 0 _B The output COUT63 is in the passive state. 1 _B The output COUT63 is enabled for the PWM signal generated by T13.
0	6	r	reserved; returns 0 if read; should be written with 0;

Capture/Compare Unit 6 (CCU6)

20.8.2 Trap Control Register

The register TRPCTRL/H controls the trap functionality. It contains independent enable bits for each output signal and control bits to select the behavior in case of a trap condition. The trap condition is a low level on the $\overline{\text{CTRAP}}$ input pin, that is monitored (inverted level) by bit ISH.TRPF. While TRPF=1 (trap input active), the trap state bit IS.TRPS is set to 1.

TRPCTRL

Trap Control Register Low

(FE_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0					TRPM2	TRPM1	TRPM0
r					rw	rw	rw

Field	Bits	Type	Description
TRPM1, TRPM0	1, 0	rw	Trap Mode Control Bits 1, 0 These two bits define the behavior of the selected outputs when leaving the trap state after the trap condition has become inactive again. A synchronization to the timer driving the PWM pattern avoids unintended pulses when leaving the trap state. The combination [TRPM1, TRPM0] leads to: 00 _B The trap state is left (return to normal operation) after TRPF has become 0 again when a zero-match of T12 (while counting up) is detected (synchronization to T12). 01 _B The trap state is left (return to normal operation) after TRPF has become 0 again when a zero-match of T13 is detected (synchronization to T13). 10 _B reserved 11 _B The trap state is left (return to normal operation) immediately after TRPF has become 0 again without any synchronization to T12 or T13.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
TRPM2	2	rw	Trap Mode Control Bit 2 This bit defines how the trap flag TRPF can be cleared after the trap input condition ($\overline{\text{CTRAP}} = 0$ and $\text{TRPPEN} = 1$) is no longer valid (either by $\overline{\text{CTRAP}} = 1$ or by $\text{TRPPEN} = 0$). 0_B Automatic Mode: Bit TRPF is cleared by HW if the trap input condition is no longer valid. 1_B Manual Mode: Bit TRPF stays 0 after the trap input condition is no longer valid. It has to be cleared by SW by writing $\text{ISR.RTRPF} = 1$.
0	[7:3]	r	reserved; returns 0 if read; should be written with 0;

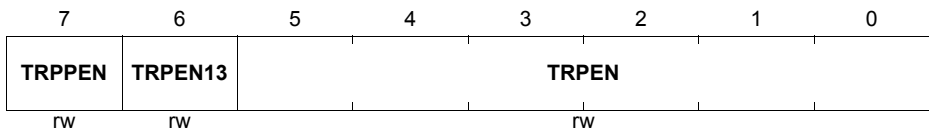
TRPCTRH

Trap Control Register High

 (FF_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2



Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
TRPEN	[5:0]	rw	Trap Enable Control Setting a bit enables the trap functionality for the following corresponding output signals: TRPEN0 = TRPCTR.8 for output CC60 TRPEN1 = TRPCTR.9 for output COUT60 TRPEN2 = TRPCTR.10 for output CC61 TRPEN3 = TRPCTR.11 for output COUT61 TRPEN4 = TRPCTR.12 for output CC62 TRPEN5 = TRPCTR.13 for output COUT62 0 _B The trap functionality of the corresponding output signal is disabled. The output state is independent from bit IS.TRPS. 1 _B The trap functionality of the corresponding output signal is enabled. The output state is set to the passive while IS.TRPS=1.
TRPEN13	6	rw	Trap Enable Control for Timer T13 0 _B The trap functionality for output COUT63 is disabled. The output state is independent from bit IS.TRPS. 1 _B The trap functionality for output COUT63 is enabled. The output state is set to the passive while IS.TRPS=1.
TRPPEN	7	rw	Trap Pin Enable This bit enables the input (pin) function for the trap generation. An interrupt can <u>only be generated</u> if a falling edge is detected at pin CTRAP while TRPPEN = 1. 0 _B The CCU6 trap functionality based on the input CTRAP is disabled. A CCU6 trap can only be generated by SW by setting bit TRPF. 1 _B The CCU6 trap functionality based on the input CTRAP is enabled. A CCU6 trap can be <u>generated</u> by SW by setting bit TRPF or by CTRAP=0.

Capture/Compare Unit 6 (CCU6)

20.8.3 Passive State Level Register

Register PSLR defines the passive state level of the PWM outputs of the module. The passive state level is the value that is driven during the passive state of the output. During the active state, the corresponding output pin drives the active state level, that is the inverted passive state level. The passive state level permits to adapt the driven output levels to the driver polarity (inverted, not inverted) of the connected power stage. The bits in this register have shadow bit fields to permit a concurrent update of all PWM-related parameters (bit field PSL is updated with T12_ST, whereas PSL63 is updated with T13_ST). The actually used values can be read (attribute "rh"), whereas the shadow bits can only be written (attribute "w").

PSLR

Passive State Level Register

(A6_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
PSL63	0	PSL					
rwh	r	rwh					

Field	Bits	Type	Description
PSL	[5:0]	rwh	Compare Outputs Passive State Level These bits define the passive level driven by the module outputs during the passive state. PSL0 = PSLR.0 for output CC60 PSL1 = PSLR.1 for output COUT60 PSL2 = PSLR.2 for output CC61 PSL3 = PSLR.3 for output COUT61 PSL4 = PSLR.4 for output CC62 PSL5 = PSLR.5 for output COUT62 0 _B The passive level is 0. 1 _B The passive level is 1.
PSL63	7	rwh	Passive State Level of Output COUT63 This bit defines the passive level driven by the module output COUT63 during the passive state. 0 _B The passive level is 0. 1 _B The passive level is 1.
0	6	r	reserved; returns 0 if read; should be written with 0;

Capture/Compare Unit 6 (CCU6)

20.8.4 Multi-Channel Mode Registers

Register MCMCTR contains control bits for the multi-channel functionality.

MCMCTR

Multi-Channel Mode Control Register (A7_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0	SWSYN			0	SWSEL		
r	rw			r	rw		

Field	Bits	Type	Description
SWSEL	[2:0]	rw	Switching Selection Bit field SWSEL selects one of the following trigger request sources (next multi-channel event) for the shadow transfer MCM_ST from MCMPS to MCMP. The trigger request is stored in the reminder flag R until the shadow transfer is done and flag R is cleared automatically with the shadow transfer. The shadow transfer takes place synchronously with an event selected in bit field SWSYN. 000 _B No trigger request will be generated 001 _B Correct Hall pattern detected (CM_CHE) 010 _B T13 period-match detected (while counting up) 011 _B T12 one-match (while counting down) 100 _B T12 channel 1 compare-match detected (phase delay function) 101 _B T12 period match detected (while counting up) 110 _B reserved, no trigger request will be generated 111 _B reserved, no trigger request will be generated

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
SWSYN	[5:4]	rw	Switching Synchronization Bit field SWSYN defines the synchronization mechanism of the shadow transfer event MCM_ST if it has been requested before (flag R set by an event selected by SWSEL) and if MCMEN = 1. This feature permits the synchronization of the outputs to the PWM source, that is used for modulation (T12 or T13). 00 _B Direct; the trigger event immediately leads to the shadow transfer 01 _B A T13 zero-match triggers the shadow transfer 10 _B A T12 zero-match (while counting up) triggers the shadow transfer 11 _B reserved; no action
0	[7:6], 3	r	reserved; returns 0 if read; should be written with 0;

Capture/Compare Unit 6 (CCU6)

Register MCMOUTSL/H contains bits used as pattern input for the multi-channel mode and the Hall mode. This register is a shadow register (that can be read and written) for register MCMOUT, indicating the currently active signals.

MCMOUTSL

Multi-Channel Mode Output Shadow Register Low

(9E_H)
Reset Value: 00_H
RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
STRMCM	0	MCMP5					
w	r	rw					

Field	Bits	Type	Description
MCMP5	[5:0]	rw	Multi-Channel PWM Pattern Shadow Bit field MCMP5 is the shadow bit field for bit field MCMP. The multi-channel shadow transfer is triggered by MCM_ST according to the transfer conditions defined by register MCMCTR.
STRMCM	7	w	Shadow Transfer Request for MCMP5 Writing STRMCM = 1 leads to an immediate activation of MCM_ST to update bit field MCMP by the value of MCMP5. When read, this bit always delivers 0. 0 _B No action. 1 _B Bit field MCMP is updated.
0	6	r	reserved; returns 0 if read; should be written with 0;

MCMOUTSH

Multi-Channel Mode Output Shadow Register High

(9F_H)
Reset Value: 00_H
RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
STRHP	0	CURHS			EXPHS		
rw	r	rw			rw		

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
EXPHS	[2:0]	rw	Expected Hall Pattern Shadow Bit field EXPHS is the shadow bit field for bit field EXPH. The shadow transfer takes place when a correct Hall event is detected (CM_CHE).
CURHS	[5:3]	rw	Current Hall Pattern Shadow Bit field CURHS is the shadow bit field for bit field CURH. The shadow transfer takes place when a correct Hall event is detected (CM_CHE).
STRHP	7	w	Shadow Transfer Request for the Hall Pattern Writing STRHP = 1 leads to an immediate activation of HP_ST to update bit fields EXPH and CURH by EXPHS and CURHS. When read, this bit always delivers 0. 0 _B No action. 1 _B Bit fields EXPH and CURH are updated.
0	6	r	reserved; returns 0 if read; should be written with 0;

MCMOUTL

Multi-Channel Mode Output Register Low

(9A_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	R						
r	rh				rw		

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
MCMP	[5:0]	rh	<p>Multi-Channel PWM Pattern</p> <p>Bit field MCMP defines the output pattern for the multi-channel mode. If this mode is enabled by MODCTR.MCMEN = 1, the output state of all T12 related PWM outputs can be modified.</p> <p>This bit field is 0 while IS.IDLE = 1.</p> <p>MCMP0 = MCMOUT.0 for output CC60</p> <p>MCMP1 = MCMOUT.1 for output COUT60</p> <p>MCMP2 = MCMOUT.2 for output CC61</p> <p>MCMP3 = MCMOUT.3 for output COUT61</p> <p>MCMP4 = MCMOUT.4 for output CC62</p> <p>MCMP5 = MCMOUT.5 for output COUT62</p> <p>0_B The output is set to the passive state. A PWM generated by T12 or T13 are not taken into account.</p> <p>1_B The output can be in the active state, depending on the enabled PWM modulation signals generated by T12, T13 and the trap state.</p>
R	6	rh	<p>Reminder Flag</p> <p>This flag indicates that the shadow transfer from MCMPS to MCMP has been requested by the selected trigger source. It is cleared when the shadow transfer takes place or while MCMEN=0.</p> <p>0_B A shadow transfer MCM_ST is not requested.</p> <p>1_B A shadow transfer MCM_ST is requested, but has not yet been executed, because the selected synchronization condition has not yet occurred.</p>
0	6	r	<p>reserved;</p> <p>returns 0 if read; should be written with 0;</p>

Capture/Compare Unit 6 (CCU6)

MCMOUTH

Multi-Channel Mode Output Register High

(9B_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	CURH				EXPH		
r	rw				rw		

Field	Bits	Type	Description
EXPH	[10:8]	rh	Expected Hall Pattern Bit field EXPH is updated by a shadow transfer HP_ST from bit field EXPHS. If HCRDY = 1, EXPH is compared to the sampled CCPOSx inputs in order to detect the occurrence of the next desired (=expected) hall pattern or a wrong pattern. If the sampled hall pattern at the hall input pins is equal to bit field EXPH, a correct Hall event has been detected (CM_CHE).
CURH	[13:11]	rh	Current Hall Pattern Bit field CURH is updated by a shadow transfer HP_ST from bit field CURHS. If HCRDY = 1, CURH is compared to the sampled CCPOSx inputs in order to detect a spike. If the sampled Hall pattern at the Hall input pins is equal to bit field CURH, no Hall event has been detected. If the sampled Hall input pattern is neither equal to CURH nor equal to EXPH, the Hall event was not the desired one and may be due to a fatal error (e.g. blocked rotor, etc.). In this case, a wrong Hall event has been detected (CM_WHE).
0	[7:6]	r	reserved; returns 0 if read; should be written with 0;

20.9 Interrupt Handling

This section describes the interrupt handling of the CCU6 module.

20.9.1 Interrupt Structure

The HW interrupt event or the SW setting of the corresponding interrupt set bit (in register ISS) sets the event indication flags (in register IS) and can trigger the interrupt generation. The interrupt pulse is generated independently from the interrupt status flag in register IS (it is not necessary to clear the related status bit to be able to generate another interrupt). The interrupt flag can be cleared by SW by writing to the corresponding bit in register ISR.

If enabled by the related interrupt enable bit in register IEN, an interrupt pulse can be generated on one of the four service request outputs (SR0 to SR3) of the module. If more than one interrupt source is connected to the same interrupt node pointer (in register INP), the requests are logically OR-combined to one common service request output (see [Figure 20-41](#)).

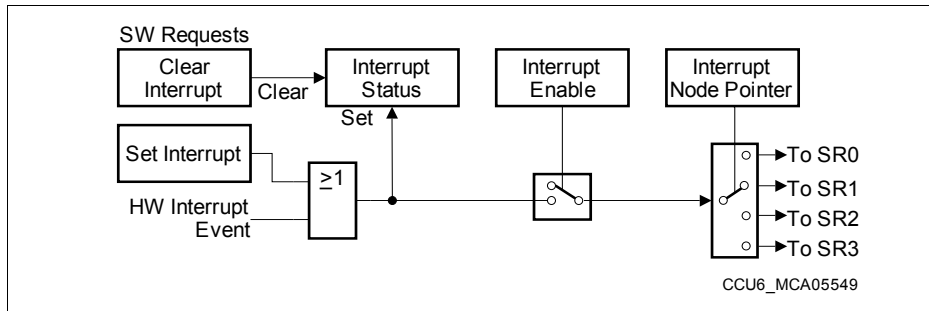


Figure 20-41 General Interrupt Structure

The available interrupt events in the CCU6 are shown in [Figure 20-42](#).

Capture/Compare Unit 6 (CCU6)

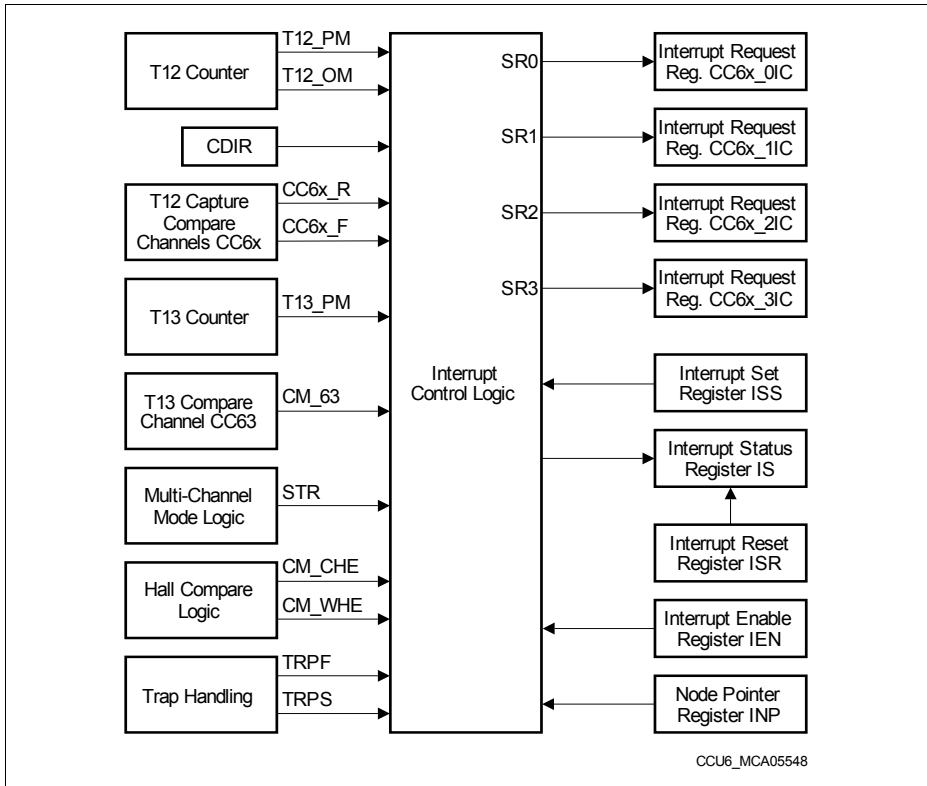


Figure 20-42 Interrupt Sources and Events

20.9.2 Interrupt Registers

20.9.2.1 Interrupt Status Register

Register ISL/H contains the individual interrupt request bits. This register can only be read, write actions have no impact on the contents of this register. The SW can set or clear the bits individually by writing to the registers ISSL/H (to set the bits) or to register ISRL/H (to clear the bits).

The interrupt generation is independent from the value of the bits in register ISL/H, e.g. the interrupt will be generated (if enabled) even if the corresponding bit is already set. The trigger for an interrupt generation is the detection of a set condition (by HW or SW) for the corresponding bit in register ISL/H.

In compare mode (and hall mode), the timer-related interrupts are only generated while the timer is running ($T1xR=1$). In capture mode, the capture interrupts are also generated while the timer T12 is stopped.

Note: Not all bits in register ISL/H can generate an interrupt. Other status bits have been added, that have a similar structure for their set and clear actions. It is recommended that SW checks the interrupt bits bit-wisely (instead of common OR over the bits).

ISL

Interrupt Status Register Low

(9C_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
T12PM	T12OM	ICC62F	ICC62R	ICC61F	ICC61R	ICC60F	ICC60R
rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
ICC60R, ICC61R, ICC62R	0, 2, 4	rh	Capture, Compare-Match Rising Edge Flag This bit indicates that event CC6x_R has been detected. This event occurs in compare mode when a compare-match is detected while T12 is counting up (CM_6x and CDIR = 0) and in capture mode when a rising edge is detected at the related input CC6xIN. 0 _B The event has not yet been detected. 1 _B The event has been detected.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
ICC60F, ICC61F, ICC62F	1, 3, 5	rh	Capture, Compare-Match Falling Edge Flag This bit indicates that event CC6x_F has been detected. This event occurs in compare mode when a compare-match is detected while T12 is counting down (CM_6x and CDIR = 1) and in capture mode when a falling edge is detected at the related input CC6xIN. 0 _B The event has not yet been detected. 1 _B The event has been detected.
T12OM	6	rh	Timer T12 One-Match Flag This bit indicates that a timer T12 one-match while counting down (T12_OM and CDIR = 1) has been detected. 0 _B The event has not yet been detected. 1 _B The event has been detected.
T12PM	7	rh	Timer T12 Period-Match Flag This bit indicates that a timer T12 period-match while counting up (T12_PM and CDIR = 0) has been detected. 0 _B The event has not yet been detected. 1 _B The event has been detected.

ISH
Interrupt Status Register High
RMAP: 0, PAGE: 3
(9D_H)
Reset Value: 00_H

7	6	5	4	3	2	1	0
STR	IDLE	WHE	CHE	TRPS	TRPF	T13PM	T13CM
rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
T13CM	0	rh	Timer T13 Compare-Match Flag This bit indicates that a timer T13 compare-match (CM_63) has been detected. 0 _B The event has not yet been detected. 1 _B The event has been detected.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
T13PM	1	rh	Timer T13 Period-Match Flag This bit indicates that a timer T13 period-match (T13_PM) has been detected. 0 _B The event has not yet been detected. 1 _B The event has been detected.
TRPF	2	rh	Trap Flag This bit indicates if a trap condition (input $\overline{\text{CTRAP}}$ = 0 or by SW) is / has been detected. If TRM2= 0, it becomes cleared automatically if $\overline{\text{CTRAP}}$ = 1 or TRPPEN = 0, whereas if TRM2 = 1, it has to be cleared by writing RTRPF = 1. 0 _B The trap condition has not been detected. 1 _B The trap condition is / has been detected.
TRPS	3	rh	Trap State¹⁾ This bit indicates the actual trap state. It is set if TRPF = 1 and becomes cleared according to the mode selected in register TRPCTR. 0 _B The trap state is not active. 1 _B The trap state is active.
CHE	4	rh	Correct Hall Event This bit indicates that a correct Hall event (CM_CHE) has been detected. 0 _B The event has not yet been detected. 1 _B The event has been detected.
WHE	5	rh	Wrong Hall Event This bit indicates that a wrong Hall event (CM_WHE) has been detected. 0 _B The event has not yet been detected. 1 _B The event has been detected.
IDLE	6	rh	IDLE State If enabled by ENIDLE = 1, this bit is set together with bit WHE and it has to be cleared by SW. 0 _B No action. 1 _B Bit field MCMP is cleared, the selected outputs are set to passive state.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
STR	7	rh	Multi-Channel Mode Shadow Transfer Request This bit indicates that a shadow transfer from MCMPS to MCMP (MCM_ST) has taken place. 0 _B The event has not yet been detected. 1 _B The event has been detected.

- 1) During the trap state, the selected outputs are set to the passive state. The logic level driven during the passive state is defined by the corresponding bit in register PSLR. Bits TRPS=1 and TRPF=0 can occur if the trap condition is no longer active but the selected synchronization has not yet taken place.

Capture/Compare Unit 6 (CCU6)
20.9.2.2 Interrupt Status Set Register

Register ISSL/H contains individual interrupt request set bits to generate a CCU6 interrupt request by software. Writing a 1 sets the bit(s) in register ISL/H at the corresponding bit position(s) and can generate an interrupt event (if available and enabled).

All bit positions read as 0.

ISSL
Interrupt Status Set Register Low (A4_H)
Reset Value: 00_H
RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
ST12PM	ST12OM	SCC62F	SCC62R	SCC61F	SCC61R	SCC60F	SCC60R
W	W	W	W	W	W	W	W

Field	Bits	Type	Description
SCC60R, SCC61R, SCC62R	0, 2, 4	w	Set Capture, Compare-Match Rising Edge Flag 0 _B No action 1 _B Bit CC6xR will be set.
SCC60F, SCC61F, SCC62F	1, 3, 5	w	Set Capture, Compare-Match Falling Edge Flag 0 _B No action 1 _B Bit CC6xF will be set.
ST12OM	6	w	Set Timer T12 One-Match Flag 0 _B No action 1 _B Bit T12OM will be set.
ST12PM	7	w	Set Timer T12 Period-Match Flag 0 _B No action 1 _B Bit T12PM will be set.

ISSH
Interrupt Status Set Register High (A5_H)
Reset Value: 00_H
RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
SSTR	SIDLE	SWHE	SCHE	SWHC	STRPF	ST13PM	ST13CM
W	W	W	W	W	W	W	W

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
ST13CM	0	w	Set Timer T13 Compare-Match Flag 0 _B No action 1 _B Bit T13CM will be set.
ST13PM	1	w	Set Timer T13 Period-Match Flag 0 _B No action 1 _B Bit T13PM will be set.
STRPF	2	w	Set Trap Flag 0 _B No action 1 _B Bits TRPF and TRPS will be set.
SWHC	3	w	Software Hall Compare 0 _B No action 1 _B The Hall compare action is triggered.
SCHE	4	w	Set Correct Hall Event Flag 0 _B No action 1 _B Bit CHE will be set.
SWHE	5	w	Set Wrong Hall Event Flag 0 _B No action 1 _B Bit WHE will be set.
SIDLE	6	w	Set IDLE Flag 0 _B No action 1 _B Bit IDLE will be set.
SSSTR	7	w	Set STR Flag 0 _B No action 1 _B Bit STR will be set.

Capture/Compare Unit 6 (CCU6)
20.9.2.3 Status Reset Register

Register ISRL/H contains bits to individually clear the interrupt event flags by software. Writing a 1 clears the bit(s) in register IS at the corresponding bit position(s). All bit positions read as 0.

ISRL
Interrupt Status Reset Register Low (A4_H)
Reset Value: 00_H
RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
RT12PM	RT12OM	RCC62F	RCC62R	RCC61F	RCC61R	RCC60F	RCC60R
W	W	W	W	W	W	W	W

Field	Bits	Type	Description
RCC60R, RCC61R, RCC62R	0, 2, 4	w	Reset Capture, Compare-Match Rising Edge Flag 0 _B No action 1 _B Bit CC6xR will be cleared.
RCC60F, RCC61F, RCC62F	1, 3, 5	w	Reset Capture, Compare-Match Falling Edge Flag 0 _B No action 1 _B Bit CC6xF will be cleared.
RT12OM	6	w	Reset Timer T12 One-Match Flag 0 _B No action 1 _B Bit T12OM will be cleared.
RT12PM	7	w	Reset Timer T12 Period-Match Flag 0 _B No action 1 _B Bit T12PM IS will be cleared.

ISRH
Interrupt Status Reset Register High (A5_H)
Reset Value: 00_H
RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
RSTR	RIDLE	RWHE	RCHE	0	RTRPF	RT13PM	RT13CM
W	W	W	W	r	W	W	W

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
RT13CM	0	w	Reset Timer T13 Compare-Match Flag 0 _B No action 1 _B Bit T13CM will be cleared.
RT13PM	1	w	Reset Timer T13 Period-Match Flag 0 _B No action 1 _B Bit T13PM will be cleared.
RTRPF	2	w	Reset Trap Flag 0 _B No action 1 _B Bit TRPF will be cleared (not taken into account while input CTRAP=0 and TRPPEN=1.
RCHE	4	w	Reset Correct Hall Event Flag 0 _B No action 1 _B Bit CHE will be cleared.
RWHE	5	w	Reset Wrong Hall Event Flag 1 _B No action 0 _B Bit WHE will be cleared.
RIDLE	6	w	Reset IDLE Flag 0 _B No action 1 _B Bit IDLE will be cleared.
RSTR	7	w	Reset STR Flag 0 _B No action 1 _B Bit STR will be cleared.
0	3	r	reserved; returns 0 if read; should be written with 0;

Capture/Compare Unit 6 (CCU6)

20.9.2.4 Interrupt Enable Register

Register IENL/H contains the interrupt enable bits and a control bit to enable the automatic idle function in the case of a wrong hall pattern.

IENL

Interrupt Enable Register Low

(9C_H)

Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
ENT12PM	ENT12OM	ENCC62F	ENCC62R	ENCC61F	ENCC61R	ENCC60F	ENCC60R
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENCC60R, ENCC61R, ENCC62R	0, 2, 4	rw	Capture, Compare-Match Rising Edge Interrupt Enable for Channel CC6x 0 _B No interrupt will be generated if the set condition for bit CC6xR in register IS occurs. 1 _B An interrupt will be generated if the set condition for bit CC6xR in register IS occurs. The service request output that will be activated is selected by bit field INPCC6x.
ENCC60F, ENCC61F, ENCC62F	1, 3, 5	rw	Capture, Compare-Match Falling Edge Interrupt Enable for Channel CC6x 0 _B No interrupt will be generated if the set condition for bit CC6xF in register IS occurs. 1 _B An interrupt will be generated if the set condition for bit CC6xF in register IS occurs. The service request output that will be activated is selected by bit field INPCC6x.
ENT12OM	6	rw	Enable Interrupt for T12 One-Match 0 _B No interrupt will be generated if the set condition for bit T12OM in register IS occurs. 1 _B An interrupt will be generated if the set condition for bit T12OM in register IS occurs. The service request output that will be activated is selected by bit field INPT12.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
ENT12PM	7	rw	Enable Interrupt for T12 Period-Match 0_B No interrupt will be generated if the set condition for bit T12PM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T12PM in register IS occurs. The service request output that will be activated is selected by bit field INPT12.

IENH

Interrupt Enable Register High
RMAP: 0, PAGE: 2

(9D_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
ENSTR	ENIDLE	ENWHE	ENCHE	0	ENTRPF	ENT13PM	ENT13CM
rw	rw	rw	rw	r	rw	rw	rw

Field	Bits	Type	Description
ENT13CM	0	rw	Enable Interrupt for T13 Compare-Match 0_B No interrupt will be generated if the set condition for bit T13CM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T13CM in register IS occurs. The service request output that will be activated is selected by bit field INPT13.
ENT13PM	1	rw	Enable Interrupt for T13 Period-Match 0_B No interrupt will be generated if the set condition for bit T13PM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T13PM in register IS occurs. The service request output that will be activated is selected by bit field INPT13.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
ENTRPF	2	rw	Enable Interrupt for Trap Flag 0_B No interrupt will be generated if the set condition for bit TRPF in register IS occurs. 1_B An interrupt will be generated if the set condition for bit TRPF in register IS occurs. The service request output that will be activated is selected by bit field INPERR.
ENCHE	4	rw	Enable Interrupt for Correct Hall Event 0_B No interrupt will be generated if the set condition for bit CHE in register IS occurs. 1_B An interrupt will be generated if the set condition for bit CHE in register IS occurs. The service request output that will be activated is selected by bit field INPCHE.
ENWHE	5	rw	Enable Interrupt for Wrong Hall Event 0_B No interrupt will be generated if the set condition for bit WHE in register IS occurs. 1_B An interrupt will be generated if the set condition for bit WHE in register IS occurs. The service request output that will be activated is selected by bit field INPERR.
ENIDLE	6	rw	Enable Idle This bit enables the automatic entering of the idle state (bit IDLE will be set) after a wrong hall event has been detected (bit WHE is set). During the idle state, the bit field MCMP is automatically cleared. 0_B The bit IDLE is not automatically set when a wrong hall event is detected. 1_B The bit IDLE is automatically set when a wrong hall event is detected.
ENSTR	7	rw	Enable Multi-Channel Mode Shadow Transfer Interrupt 0_B No interrupt will be generated if the set condition for bit STR in register IS occurs. 1_B An interrupt will be generated if the set condition for bit STR in register IS occurs. The service request output that will be activated is selected by bit field INPCHE.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
0	3	r	reserved; returns 0 if read; should be written with 0;

Capture/Compare Unit 6 (CCU6)

20.9.2.5 Interrupt Node Pointer Register

Register INPL/H contains the interrupt node pointers allowing a flexible interrupt handling. These bit fields define which service request output will be activated if the corresponding interrupt event occurs and the interrupt generation for this event is enabled.

INPL

Interrupt Node Pointer Register Low (9E_H)

Reset Value: 40_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
INPCHE		INPCC62		INPCC61		INPCC60	
rw		rw		rw		rw	

Field	Bits	Type	Description
INPCC60, INPCC61, INPCC62	[1:0], [3:2], [5:4]	rw	Interrupt Node Pointer for Channel CC6x Interrupts This bit field defines the service request output activated due to a set condition for bit CC6xR (if enabled by bit ENCC6xR) or for bit CC6xF (if enabled by bit ENCC6xF). 00 _B Service request output SR0 is selected. 01 _B Service request output SR1 is selected. 10 _B Service request output SR2 is selected. 11 _B Service request output SR3 is selected.
INPCHE	[7:6]	rw	Interrupt Node Pointer for the CHE Interrupt This bit field defines the service request output activated due to a set condition for bit CHE (if enabled by bit ENCHE) or for bit STR (if enabled by bit ENSTR). Coding see INPCC6x.

Capture/Compare Unit 6 (CCU6)

INPH

Interrupt Node Pointer Register High (9F_H)

Reset Value: 39_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0		INPT13		INPT12		INPERR	
r		rw		rw		rw	

Field	Bits	Type	Description
INPERR	[1:0]	rw	Interrupt Node Pointer for Error Interrupts This bit field defines the service request output activated due to a set condition for bit TRPF (if enabled by bit ENTRPF) or for bit WHE (if enabled by bit ENWHE). Coding see INPCC6x.
INPT12	[3:2]	rw	Interrupt Node Pointer for Timer12 Interrupts This bit field defines the service request output activated due to a set condition for bit T12OM (if enabled by bit ENT12OM) or for bit T12PM (if enabled by bit ENT12PM). Coding see INPCC6x.
INPT13	[5:4]	rw	Interrupt Node Pointer for Timer13 Interrupt This bit field defines the service request output activated due to a set condition for bit T13CM (if enabled by bit ENT13CM) or for bit T13PM (if enabled by bit ENT13PM). Coding see INPCC6x.
0	[7:6]	r	reserved; returns 0 if read; should be written with 0;

20.10 General Module Operation

This section provides information about the:

- Input selection (see [Section 20.10.1](#))
- General register description (see [Section 20.10.2](#))

20.10.1 Input Selection

Each CCU6 input signal can be selected from a vector of four or eight possible inputs by programming the port input select registers [PISEL0L](#), [PISEL0H](#) and [PISEL2](#). This permits to adapt the pin functionality of the device to the application requirements.

The output pins for the module output signals are chosen in the ports.

Naming convention:

The input vector CC60IN[D:A] for input signal CC60IN is composed of the signals CC60INA to CC60IND.

Note: All functional inputs of the CCU6 are synchronized to f_{CC6} before they affect the module internal logic. The resulting delay of $2/f_{CC6}$ and for asynchronous signals an additional uncertainty of $1/f_{CC6}$ have to be taken into account for precise timing calculation. An edge of an input signal can only be correctly detected if the high phase and the low phase of the input signal are both longer than $1/f_{CC6}$.

Capture/Compare Unit 6 (CCU6)

20.10.2 General Registers

20.10.2.1 Port Input Select Registers

Registers PISEL0L and PISEL0H contain bit fields selecting the actual input signal for the module inputs.

PISEL0L

Port Input Select Register Low

(9E_H)

Reset Value: 00_H

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
ISTRP		ISCC62		ISCC61		ISCC60	
rw		rw		rw		rw	

Field	Bits	Type	Description
ISCC60	[1:0]	rw	Input Select for CC60 This bit field defines the port pin or that is used for the CC60 capture input. 00 _B Input pin CC60_0. 01 _B Input pin CC60_1. 10 _B CCU6 SR2 event. 11 _B ADC channel 0 boundary limit check event.
ISCC61	[3:2]	rw	Input Select for CC61 This bit field defines the port pin that is used for the CC61 capture input signal. 00 _B Input pin CC61_0. 01 _B Input pin CC61_1. 10 _B CCU6 SR2 event. 11 _B ADC channel 1 boundary limit check event.
ISCC62	[5:4]	rw	Input Select for CC62 This bit field defines the port pin that is used for the CC62 capture input signal. 00 _B Input pin CC62_0. 01 _B Input pin CC62_1. 10 _B CCU6 SR2 event. 11 _B ADC channel 2 boundary limit check event.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
ISTRP	[7:6]	rw	Input Select for CTRAP This bit field defines the options that is used for the CTRAP input signal. MODPSEL3.CPTRAPIS is concurrently to select the 8 type of sources to trigger CTRAP. 00 _B One of the input pin for CTRAP_0, CTRAP_1, CTRAP_2 and CTRAP_3 is selected. Bit CTRAPIS bit in MODPSEL3 register are used for the individual selection. 01 _B P1 overcurrent detection output is selected. ¹⁾ 10 _B Any input source from above option 00 or 01 is selected to trigger a CTRAP. ¹⁾ 11 _B ADC channel event is selected.

1) Applicable for XC83x only. Unused for XC82x.

PISEL0H
Port Input Select Register 0 High (9F_H)
Reset Value: 00_H
RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
IST12HR		ISPOS2		ISPOS1		ISPOS0	
rw		rw		rw		rw	

Field	Bits	Type	Description
ISPOS0	[1:0]	rw	Input Select for CCPOS0 This bit field defines the input signal used as CCPOS0 input. 00 _B Input pin for CCPOS0_0. 01 _B Input pin for CCPOS0_1. 10 _B Input pin for CCPOS0_2. 11 _B ADC channel 0 boundary limit check event.

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
ISPOS1	[3:2]	rw	Input Select for CCPOS1 This bit field defines the input signal used as CCPOS1 input. 00 _B Input pin for CCPOS1_0. 01 _B Input pin for CCPOS1_1. 10 _B Unused. 11 _B ADC channel 1 boundary limit check event.
ISPOS2	[5:4]	rw	Input Select for CCPOS2 This bit field defines the the port pin that is used for the CCPOS2 input signal. 00 _B Input pin for CCPOS2_0. 01 _B Input pin for CCPOS2_1. 10 _B Unused. 11 _B ADC channel 2 boundary limit check event.
IST12HR	[7:6]	rw	Input Select for T12HR This bit field defines the input signal used as T12HR input. 00 _B Any of the input pin for T12HR[7:0] ¹⁾ . 01 _B CCU6 SR2 output. 10 _B CCU6 SR3 output. 11 _B ADC channel event.

1) The selection of T12HR[7:0] can be done by bit IST12HR1 in MODIPSEL3 register from SCU module.

PISEL2
Port Input Select Register 2
(A4_H)
Reset Value: 00_H
RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0						IST13HR	
r						rw	

Capture/Compare Unit 6 (CCU6)

Field	Bits	Type	Description
IST13HR	[1:0]	rw	Input Select for T13HR This bit field defines the input signal used as T13HR input. 00 _B Any of the input pin for T13HR[7:0] ¹⁾ . 01 _B CCU6 SR2 output. 10 _B CCU6 SR3 output. 11 _B ADC channel event.
0	[7:2]	r	reserved; returns 0 if read; should be written with 0;

1) The selection of T13HR[7:0] can be done by bit IST13HR1 in MODIPSEL3 register from SCU module.

20.11 Register Mapping

The addresses of the kernel SFRs are listed in [Table 20-16](#).

Table 20-16 SFR Address List for Pages 0 – 3

Address	Page 0	Page 1	Page 2	Page 3
9AH	CC63SRL	CC63RL	T12MSELL	MCMOUTL
9BH	CC63SRH	CC63RH	T12MSELH	MCMOUTH
9CH	TCTR4L	T12PRL	IENL	ISL
9DH	TCTR4H	T12PRH	IENH	ISH
9EH	MCMOUTSL	T13PRL	INPL	PISEL0L
9FH	MCMOUTSH	T13PRH	INPH	PISEL0H
A4H	ISRL	T12DTCL	ISSL	PISEL2
A5H	ISRH	T12DTCH	ISSH	
A6H	CMPMODIFL	TCTR0L	PSLR	
A7H	CMPMODIFH	TCTR0H	MCMCTR	
FAH	CC60SRL	CC60RL	TCTR2L	T12L
FBH	CC60SRH	CC60RH	TCTR2H	T12H
FCH	CC61SRL	CC61RL	MODCTRL	T13L
FDH	CC61SRH	CC61RH	MODCTRH	T13H
FEH	CC62SRL	CC62RL	TRPCTRL	CMPSTATL
FFH	CC62SRH	CC62RH	TRPCTRH	CMPSTATH

21 Analog to Digital Converter

The Analog to Digital Converter module (ADC) of the XC82x uses the successive approximation method to convert analog input values (voltages) to discrete digital values.

One ADC kernel (ADC0) operate on a user-selectable number of input channels.

The input channels can be selected and arbitrated flexibly.

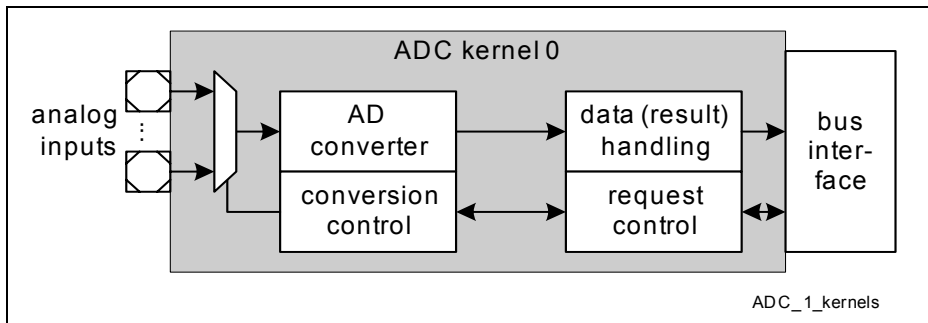


Figure 21-1 ADC Module Block Diagram

This section gives an overview about the feature of the ADC module and introduces the general structure which is described in the below format:

- [“Introduction and Basic Structure” on Page 21-7](#)
- [“Configuration of General Functions” on Page 21-13](#)
- [“Conversion Request Generation” on Page 21-17](#)
- [“Request Source Arbitration” on Page 21-38](#)
- [“Analog Input Channel Configuration” on Page 21-45](#)
- [“Conversion Result Handling” on Page 21-64](#)
- [“Interrupt Request Handling” on Page 21-83](#)
- [“Register Mapping” on Page 21-89](#)

Analog to Digital Converter

The following features describe the functionality of an ADC kernel:

- Input voltage range from 0 V up to analog supply voltage ($V_{DDP} = 3.0 \text{ V to } 5.5 \text{ V}$)
- Three internal reference voltage source selectable for each channel to support ratiometric measurements and different signal scales, which are:
 - Internal V_{DDP} and V_{SSP}
 - Internal $1.2V_{ref}$ and CH0 used as ADC voltage reference ground¹⁾
 - Internal $1.2V_{ref}$ and V_{SSP} ¹⁾
- Up to 4 analog input channels
- Conversion speed and sample time adjustable to adapt to sensors and reference
- Conversion time below 1 μs (depending on result width and sample time)
- Flexible source selection and arbitration
 - Single-channel conversion (single or repeated)
 - Configurable auto scan conversions (single or repeated)
 - Programmable arbitrary conversion sequence (single or repeated)
 - Conversions triggered by software, timer events, or external events
 - Wait-for-start mode for maximum throughput or
 - Cancel-inject-restart mode for reduced conversion delay
- Powerful result handling
 - Selectable result width of 8 to 10 bits
 - 4 independent result registers
 - Configurable limit checking against programmable border values
 - Data rate reduction through adding a selectable number of conversion results
 - First order digital low pass filter through averaging of the conversion results
- Flexible interrupt generation based on selectable events
- Support of power saving modes
- Additional features
 - Out of range (ORC) voltage comparator detection for each input channel that is able to trigger other modules
 - Configurable limit checker able to trigger other modules

1) A minimum of 50 μsec is needed between conversions when using this mode.

21.1 System Information

This section provides system information relevant to the ADC.

21.1.1 Pinning

The ADC pin assignment for XC82x is shown in [Table 21-1](#).

Table 21-1 ADC Pin Functions and Selection

Pin	Function	Description	Selected By
P2.0	CH0	Analog input channel 0	P2_EN.P0 = 1 _B
P2.1	CH1	Analog input channel 1	P2_EN.P1 = 1 _B
P2.2	CH2	Analog input channel 2	P2_EN.P2 = 1 _B
P2.3	CH3	Analog input channel 3	P2_EN.P3 = 1 _B

21.1.2 Clocking Configuration

The ADC kernel runs on the FPCLK at a fixed frequency of 48 MHz. See [Section 21.5.1](#)

If the ADC functionality is not required at all, it can be completely disabled by gating off its clock input for maximal power reduction. This is done by setting bit ADC_DIS in register PMCON1 as described below.

The bit field PAGE of SCU_PAGE register must be programmed before accessing the PMCON1 register.

PMCON1

Peripheral Management Control Register 1(EF_H)

Reset Value: DF_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	0	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	r	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ADC_DIS	0	rw	ADC Disable Request. Active high. 0 ADC is in normal operation. 1 Request to disable the ADC. (default)
0	5	r	Reserved Returns 0 if read; should be written with 0.

21.1.3 Interrupt Events and Assignment

Table 21-2 lists the interrupt event sources from the ADC, and the corresponding event interrupt enable bit and flag bit.

Table 21-2 ADC Interrupt Events

Event	Event Interrupt Enable Bit	Event Flag Bit	Service Request Output
Request Source Event Interrupts	ADC_Q0R0.ENS1	ADC_EVINFR.EVINFx	SR0
	ADC_QBUR0.ENS1		
Channel Event Interrupts	ADC_GLOBCTR.CLCIEN	ADC_CHINFR.CHINFRx	SR0
Result Event Interrupts	ADC_RCRx (x = 0 - 3).IEN	ADC_EVINFR.EVINFRx	SR0
Out of Range Comparator Event Interrupts	ADC_GLOBCTR.ORCIEN	ADC_LORC.LOREx	SR1

Table 21-3 shows the interrupt node assignment for each ADC interrupt source.

Table 21-3 ADC Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
SR0	IEN1.EADC	IRCON1.ADCSR0	33 _H
SR1		IRCON1.ADCSR1	

21.1.4 IP Interconnection

The ADC has interconnection to other peripherals enabling higher level of automation without requiring software. **Table 21-4** describes the interconnection from ADC outputs, channel events and out of range comparator events, to CCU6 and Timer 2 inputs. **Table 21-5** describes the interconnection from CCU6 and LEDTSCU outputs to the external request trigger input of ADC.

Table 21-4 ADC Output Interconnections

ADC Function/Signal	Connected Other Module Inputs	Selected By
Channel Events		
ADC channel event (o): ADC_CHEV	CCU6 input (i): T12HR	CCU6_PISEL0H.IST12HR = 11 _B
	CCU6 input (i): T13HR	CCU6_PISEL2.IST13HR = 11 _B
ADC channel event 0 (o): ADC_CHEV0	CCU6 input (i): CTRAP	CCU6_PISEL0L.ISTRP = 11 _B
ADC channel event 1 (o): ADC_CHEV1	No connection	-
ADC channel event 2 (o): ADC_CHEV2	No connection	-
ADC boundary event 0 (o): ADC_BF0	CCU6 input (i): CC60	CCU6_PISEL0L.ISCC60 = 11 _B
	CCU6 input (i): CCPOS0	CCU6_PISEL0H.ISPOS0 = 11 _B
ADC boundary event 1 (o): ADC_BF1	CCU6 input (i): CC61	CCU6_PISEL0L.ISCC61 = 11 _B
	CCU6 input (i): CCPOS1	CCU6_PISEL0H.ISPOS1 = 11 _B
ADC boundary event 2 (o): ADC_BF2	CCU6 input (i): CC62	CCU6_PISEL0L.ISCC62 = 11 _B
	CCU6 input (i): CCPOS2	CCU6_PISEL0H.ISPOS2 = 11 _B
Out of Range Comparator (ORC)		
ORC event 0 (o): ORCEVENT0	CCU6 input (i): T12HR	MODPISEL3.IST12HR1 = 001 _B CCU6_PISEL0H.IST12HR = 00 _B
	Timer 2 input (i): T2EX	MODPISEL2.T2EXIS = 100 _B
ORC event 1 (o): ORCEVENT1	Timer 2 input (i): T2EX	MODPISEL2.T2EXIS = 101 _B
ORC event 2 (o): ORCEVENT2	CCU6 input (i): T12HR	MODPISEL3.IST12HR1 = 100 _B CCU6_PISEL0H.IST12HR = 00 _B
	CCU6 input (i): T13HR	MODPISEL3.IST13HR1 = 100 _B CCU6_PISEL2.IST13HR = 00 _B
ORC event 3 (o): ORCEVENT3	CCU6 input (i): CTRAP	MODPISEL3.CTRAPIS = 11 _B CCU6_PISEL0L.ISTRP = 00 _B

Table 21-5 ADC Input Interconnection

ADC Function/Signal	Connected Other Module Function/Signal
Request Source x (x = 0,1)	
Request Source x Trigger 0 Input (i): REQTRxA	CCU6 service request output SR2 (o): CCU6_SR2
Request Source x Trigger 1 Input (i): REQTRxB	CCU6 service request output SR3 (o): CCU6_SR3
Request Source x Trigger 2 Input (i): REQTRxC	CCU6 T12 period match (o): T12PM
Request Source x Trigger 3 Input (i): REQTRxD	CCU6 T13 period match (o): T13PM
Request Source x Trigger 4 Input (i): REQTRxE	CCU6 T13 compare match (o): T13CM
Request Source x Trigger 5 Input (i): REQTRxF	CCU6 MCM shadow transfer (o): MCM_ST
Request Source x Trigger 6 Input (i): REQTRxG	LEDTSKU Compare match (o): LEDTS_CM
Request Source x Trigger 7 Input (i): REQTRxH	LEDTSKU Time slice interrupt (o): LEDTS_TSI

21.2 Introduction and Basic Structure

A set of functional units can be configured according to the requirements of a given application. These units build a path from the input signals to the digital results.

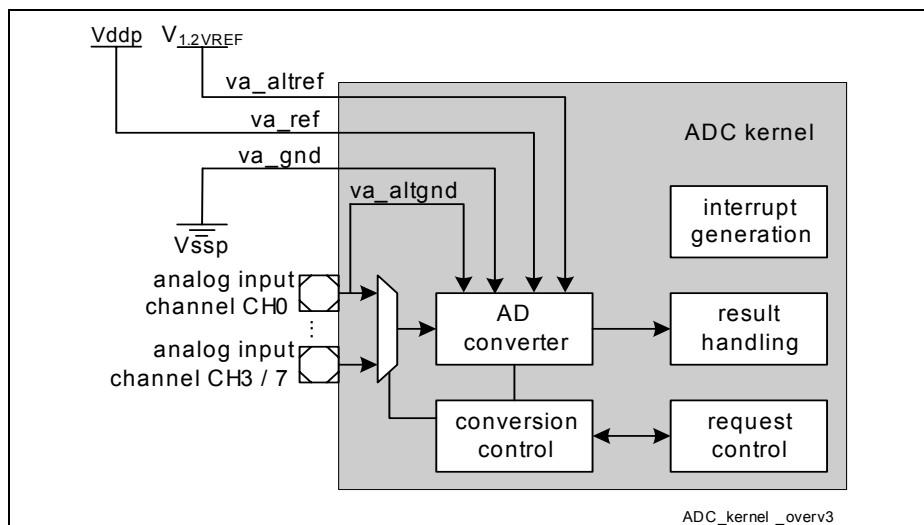


Figure 21-2 ADC Kernel Block Diagram

Request Source Control

Concurrent conversion requests can be generated by up to 2 request sources:

- A linear sequence source requests auto scan conversions of a configurable sequence of up to 8 channels
- An arbitrary sequence source requests queued conversions of up to 4 arbitrarily selectable channels

Each source can request its selected conversion sequence once or repeatedly. Each source can be enabled separately and can be triggered by external events, such as edges of PWM or timer signals, or pin transitions.

An arbiter resolves concurrent conversion requests from different sources. Requests with higher priority can either cancel a running lower-priority conversion (cancel-inject-repeat mode) or be converted immediately after the currently running conversion (wait-for-start mode). If the target result register has not been read, a conversion can be deferred (wait-for-read mode).

Analog to Digital Converter

Input Channel Selection

The analog input multiplexer selects one of up to 8 analog inputs (CH0 - CH7) to be converted. Two sources can select a linear sequence or an arbitrary sequence. The priorities of these sources can be configured.

Note: Not all analog input channels are necessarily available in all packages, due to pin limitations. Please refer to the implementation description in [Section 21.1](#).

Conversion Control

Conversion parameters, such as sample phase duration, can be configured in the input classes.

The input channels can, thus, be adjusted to the type of sensor (or other analog sources) connected to the ADC.

Analog/Digital Converter

The selected input channel is converted to a digital value by first sampling the voltage on the selected input and then generating the selected number of result bits.

Result Handling

The conversion results of each analog input channel can be directed to one of 4 result registers to be stored there. A result register can be used by a group of channels or by a single channel.

The wait-for-read mode avoids data loss due to result overwrite by blocking a conversion until the previous result has been read.

Data reduction (e.g. for digital anti-aliasing filtering) can automatically add up to 2 conversion results before interrupting the CPU.

Also, result registers can be concatenated to build FIFO structures that store a number of conversion results without overwriting previous data. This increases the allowed CPU latency for retrieving conversion data from the ADC.

A digital first order low pass filter is implemented that can continuously filter the conversion results before it is written to the result register.

Interrupt Generation

Several ADC events can issue interrupt requests to the CPU:

- **Source events** indicate the completion of a conversion sequence in the corresponding request source. This event can be used to trigger the setup of a new sequence.
- **Channel events** indicate the completion of a conversion for a certain channel. This can be combined with limit checking, so interrupt are generated only if the result is within a defined range of values.

Analog to Digital Converter

- **Result events** indicate the availability of new result data in the corresponding result register. If data reduction or digital low pass filter mode is active, events are generated only after a complete accumulation sequence.
- Out of range comparator events indicate that a voltage higher or lower than V_{ddp} is detected at the ADC input channels.

All interrupt request is assigned to two interrupt nodes.

Additional Features

Detect and trigger mechanisms are supported with two mechanisms to ensure a fast response without CPU intervention.

Out of Range Comparator (ORC) is build into every ADC channel which will trigger other modules or an interrupt when voltage out of range condition occurs. This happens when voltage at the input channel rises to above V_{ddp} level or when the input channel falls to a voltage below V_{ddp} level.

The out of range comparator is connected to other modules e.g CCU6, timer such that it is able to trigger the start or stop of other modules. All out of range comparator events is assigned to one interrupt node.

Configurable Limit Checker checks the conversion results against programmed values which will trigger other modules or an interrupt when the trigger condition is fulfilled.

The configurable limit checker is conected to other module i.e CCU6 such that is is able to trigger the start or stop of the other module.

21.3 Electrical Models

Each conversion of an analog input voltage to a digital value consists of two consecutive phases:

- During the sample phase, the input voltage is sampled and stored.
The input signal path is a simplified model for this.
- During the conversion phase the stored voltage is converted to a digital result.
The reference voltage path is a simplified model for this.

Input Signal Path

The ADC of the XC82x uses a switched capacitor field represented by C_{AIN} (small parasitic capacitances are present at each input pin). During the sample phase, the capacitor field C_{AIN} is connected to the selected analog input CHx via the input multiplexer (modeled by ideal switches and series resistors R_{AIN}).

The switch to CHx is closed during the sample phase and connects the capacitor field to the input voltage V_{AINx} .

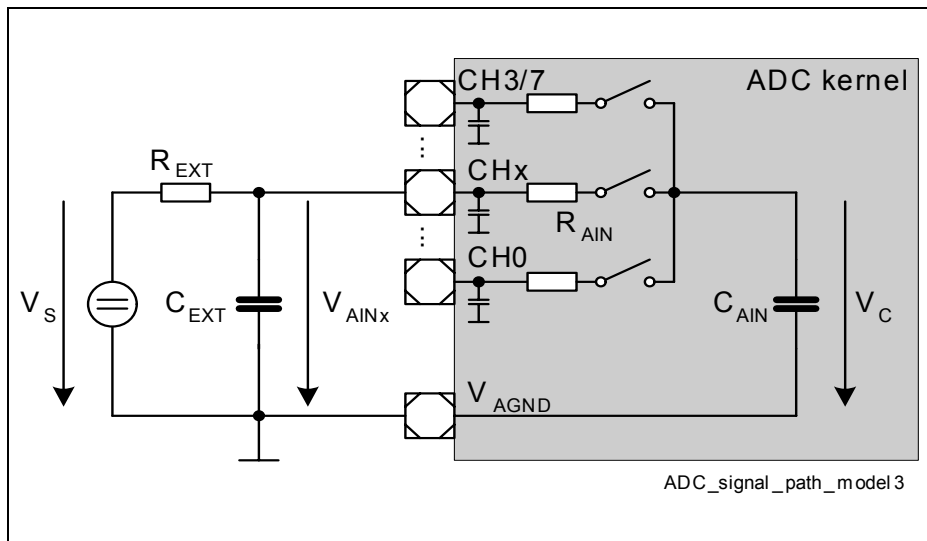


Figure 21-3 Signal Path Model

A simplified model for the analog input signal path is given in [Figure 21-3](#). An analog voltage source (value V_S) with an internal impedance of R_{EXT} delivers the analog input that should be converted.

During the sample phase the corresponding switch is closed and the capacitor field C_{AIN} is charged. Due to the low-pass behavior of the resulting RC combination, the voltage

Analog to Digital Converter

V_C to be actually converted does not immediately follow V_S . The value R_{EXT} of the analog voltage source and the desired precision of the conversion strongly define the required length of the sample phase.

To reduce the influence of R_{EXT} and to filter input noise, it is recommended to introduce a fast external blocking capacitor C_{EXT} at the analog input pin of the ADC. Like this, mainly C_{EXT} delivers the charge during the sample phase. This structure allows a significantly shorter sample phase than without a blocking capacitor, because the low-pass time constant defining the sample time is mainly given by the values of R_{AIN} and C_{AIN} .

Additionally, the capacitor C_{AIN} is automatically precharged to a voltage of approximately the half of the standard reference voltage V_{AREF} to minimize the average difference between V_{AINx} and V_C at the beginning of a sample phase. Due to varying parameters and parasitic effects, the precharge voltage of C_{AIN} is typically smaller than $V_{AREF} / 2$.

On the other hand, the charge redistribution between C_{EXT} and C_{AIN} leads to a voltage change of V_{AINx} during the sample phase. In order to keep this voltage change lower than 1 LSB_n , it is recommended to use an external blocking capacitor C_{EXT} in the range of at least $2^n \times C_{AIN}$.

The resulting low-pass filter of R_{EXT} and C_{EXT} should be dimensioned in a way to allow V_{AINx} to follow V_S between two sample phases of the same analog input channel.

Please note that, especially at high temperatures, the analog input structure of an ADC can lead to a leakage current and introduces an error due to a voltage drop over R_{EXT} . The ADC input leakage current increases if the input voltage level is close to the analog supply ground V_{SS} or to the analog power supply V_{DDP} . It is recommended to use an operating range for the input voltage between approximately 3% and 97% of V_{DDP} to reduce the input leakage current of the respective ADC channel.

Furthermore, the leakage is influenced by an overload condition at adjacent analog inputs. During an overload condition, an input voltage exceeding the supply range is applied at an input and the built-in protection circuit limits the resulting input voltage. This leads to an overload current through the protection circuit that is translated (by a coupling factor) into an additional leakage at adjacent inputs.

21.4 Transfer Characteristics and Error Definitions

The transfer characteristic of the ADC describes the association of analog input voltages to the 2^n discrete digital result values (n bits resolution). Each digital result value (in the range of 0 to 2^n-1) represents an input voltage range defined by the reference voltage range divided by 2^n . This range (called quantization step or code width) represents the granularity (called LSB_n) of the ADC. The discrete character of the digital result generates a system-inherent quantization uncertainty of $\pm 0.5 \text{ LSB}_n$ for each conversion result.

The ideal transfer curve has the first digital transition (between 0 and 1) when the analog input reaches 0.5 LSB_n . The quantization steps are equally distributed over the input voltage range.

Analog input voltages below or above the reference voltage limits lead to a saturation of the digital result at 0 or 2^n-1 .

The real transfer curve can exhibit certain deviations from the ideal transfer curve:

- The **offset error** is the deviation of the real transfer line from the ideal transfer line at the lowest code. This refers to best-fit lines through all possible codes, for both cases.
- The **gain error** is the deviation of the slope of the real transfer line from the slope of the ideal transfer line. This refers to best-fit lines through all possible codes, for both cases.
- The **differential non-linearity error** (DNL) is the deviation of the real code width (variation of the analog input voltage between two adjacent digital conversion results) from the ideal code width. A DNL value of -1 LSB_n indicates a missing code.
- The **integral non-linearity error** (INL) is the deviation of the real transfer curve from an adjusted ideal transfer curve (same offset and gain error as the real curve, but equal code widths).
- The **total unadjusted error** (TUE) describes the maximum deviation between a real conversion result and the ideal transfer characteristics over a given measurement range. Since some of these errors noted above can compensate each other, the TUE value generally is much less than the sum of the individual errors.

The TUE also covers production process variations and internal noise effects (if switching noise is generated by the system, this generally leads to an increased TUE value).

21.5 Configuration of General Functions

While many parameters can be selected individually for each channel, source, etc, some adjustments are valid for the whole ADC kernel.

21.5.1 General Clocking Scheme and Control

The different parts of an ADC kernel are driven by clock signals that are based on the clock f_{ADC} of the bus that is used to access the ADC module. The ADC in the XC82x device are connected to the system clock, so $f_{ADC} = f_{SYS}$.

- The analog clock f_{ADCI} is used as internal clock for the converter and defines the conversion length and the sample time.
See [Section 21.8.6](#).
- The digital clock f_{ADCD} is used for the arbiter and defines the duration of an arbiter round
- All other digital structures (such as interrupts, etc.) are directly driven by the module clock f_{ADC} .

Timing parameters are programmed in register [ADC_GLOBCTR](#).

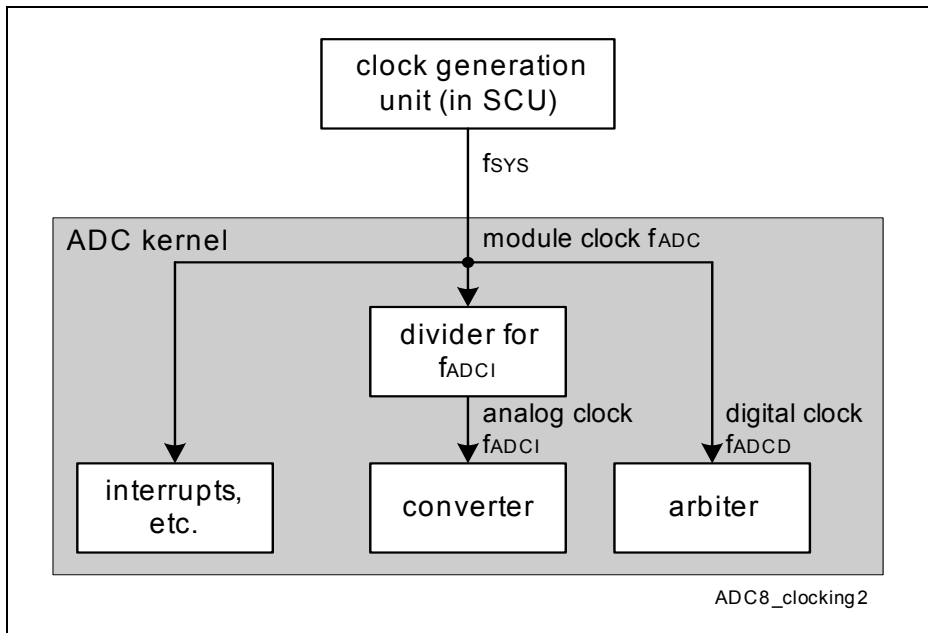


Figure 21-4 Clocking Scheme

Analog to Digital Converter

Note: If the clock generation for the converter of the ADC falls below a minimum value or is stopped during a running conversion, the conversion result can be corrupted. For correct ADC results, the frequency of f_{ADCI} must not exceed the defined range. Please, refer to the range indicated in the respective Data Sheet.

The Global Control Register defines the basic timing parameters and the basic operating mode of the converter unit, it contains bits for:

- Enabling/Disabling of the analog converter
- Defining the result bits resolution, 8/10bits wide
- Defining the divider ratio CTC for f_{ADCI} , internal clock frequency for the analog part
- Enabling/Disabling of the Out of range comparator interrupt
- Enabling/Disabling of the Channel limit checking interrupt

ADC_GLOBCTR

Global Control Register

(CA_H)

Reset Value: 30_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
ANON	DW	CTC	ORCIEN	CLCIEN	0		
rw	rw	rw	rw	rw	rw	r	

Field	Bits	Type	Description
CLCIEN	2	rw	Channel Limit Checking Interrupt Enable This bit enables the channel event interrupt related to the limit checking, see Figure 21-15 . A channel event interrupt is generated when this interrupt enable is set to "1", there is new result in buffer and result triggers the limit check unit. 0 _B The event interrupt is disabled. 1 _B The event interrupt is enabled.
ORCIEN	3	rw	Out of Range Comparator Interrupt Enable 0 _B Out of range comparator interrupt disabled. 1 _B Out of range comparator interrupt enabled.

Analog to Digital Converter

Field	Bits	Type	Description
CTC	[5:4]	rw	Conversion Time Control This bit field defines the divider ratio for the divider stage of the internal analog clock f_{ADCI} . This clock provides the internal time base for the conversion and sample time calculations. $00_B \quad f_{ADCI} = 1/3 \times f_{ADCA}$ $01_B \quad f_{ADCI} = 1/4 \times f_{ADCA}$ $10_B \quad f_{ADCI} = 1/5 \times f_{ADCA}$ $11_B \quad f_{ADCI} = 1/6 \times f_{ADCA}$ (default)
DW	6	rw	Data Width This bit field defines how many bits are converted for the result. 0_B The result is 10-bits wide (default). 1_B The result is 8-bits wide.
ANON	7	rw	Analog Part Switched On This bit enables the analog part of the ADC module and defines its operation mode. 0_B The analog part is switched off and conversions are not possible. To achieve minimal power consumption, the internal analog circuitry is in its power-down state and the generation of f_{ADCI} is stopped. 1_B The analog part of the ADC module is switched on and conversions are possible. The automatic power-down capability of the analog part is disabled.
0	[1:0]	r	Reserved Returns 0 if read; should be written with 0.

The Global Status Register indicates the current status of a conversion

ADC_GLOBSTR

Global Status Register

(CB_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	CHNR			0	SAMPLE	BUSY	
r	rh			r	rh	rh	

Analog to Digital Converter

Field	Bits	Type	Description
BUSY	0	rh	Analog Part Busy This bit indicates that a conversion is currently active. 0 _B The analog part is idle. 1 _B A conversion is currently active.
SAMPLE	1	rh	Sample Phase This bit indicates that an analog input signal is currently sampled. 0 _B The analog part is not in the sampling phase. 1 _B The analog part is in the sampling phase.
CHNR	[5:3]	rh	Channel Number This bit field indicates which analog input channel is currently converted. This information is updated when a new conversion is started. <i>Note: Bit 5 is only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>
0	2, [7:6]	r	Reserved Returns 0 if read; should be written with 0.

21.6 Conversion Request Generation

The conversion request unit of each ADC kernel autonomously handles the generation of conversion requests. Two request sources can generate requests for the conversion of an analog channel. The arbiter resolves concurrent requests and selects the channel to be converted next.

Upon a trigger event, the request source requests the conversion of a certain analog input channel or a sequence of channels.

- **Software triggers** directly activate the respective request source.
- **External triggers** synchronize the request source activation with external events, such as a trigger pulse from a timer generating a PWM signal or from a port pin.

Application software selects the trigger, the channel(s) to be converted, and the request source priority. A request source can also be activated directly by software without requiring an external trigger.

The arbiter regularly scans the request sources for pending conversion requests and selects the conversion request with the highest priority. This conversion request is then forwarded to the converter to start the conversion of the requested channel.

Each request source can operate in single-shot or in continuous mode:

- **In single-shot mode**, the programmed conversion (sequence) is requested once after being triggered. A subsequent conversion (sequence) must be triggered again.
- **In continuous mode**, the programmed conversion (sequence) is automatically requested repeatedly after being triggered once.

For each request source, external triggers are generated from one of 8 selectable trigger inputs (REQTRx[H:A]).

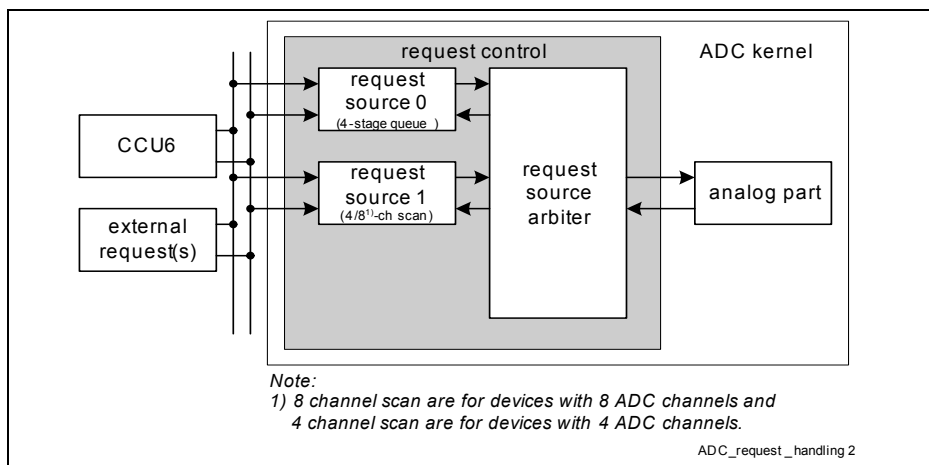


Figure 21-5 Conversion Request Unit

Analog to Digital Converter

Two types of requests sources are available:

- **A channel scan source** can issue conversion requests for a coherent sequence of input channels. This sequence begins with the highest enabled channel number and continues towards lower channel numbers. Up to channels can be enabled for the scan sequence. Each channel is converted once per sequence.

A scan source converts a series of input channels permanently or on a regular time base. For example, if programmed with low priority, some input channels can be scanned in a background task to update information that is not time-critical.

Request source 1 is a channel scan source.

- **A queued source** can issue conversion requests for an arbitrary sequence of input channels. The channel numbers for this sequence can be freely programmed. This supports application-specific conversion sequences that cannot be covered by a channel scan source. Also, multiple conversions of the same channel within a sequence are supported.

A queued source converts a series of input channels permanently or on a regular time base. For example, if programmed with medium priority, some input channels can be converted upon a specified event (e.g. synchronized to a PWM). Conversions of lower priority sources are suspended in the meantime.

Request source 0 is a 4-stage queued source.

21.6.1 Channel Scan Request Source Handling

Each analog input channel can be included in or excluded from the scan sequence (see bits in register [ADC_CRCR1](#)). The programmed register value remains unchanged by an ongoing scan sequence. The scan sequence starts with the highest enabled channel number and continues towards lower channel numbers.

Upon a load event, the request pattern is transferred to the pending bits (see register [ADC_CRCR1](#)). The pending conversion requests indicate which input channels are to be converted in an ongoing scan sequence. Each conversion start that was triggered by the scan request source, automatically clears the corresponding pending bit. If the last conversion triggered by the scan source is finished and all pending bits are cleared, the current scan sequence is considered finished and a request source event is generated.

A conversion request is only issued to the request source arbiter if at least one pending bit is set.

If the arbiter aborts a conversion triggered by the scan request source due to higher priority requests, the corresponding pending bit is automatically set. This ensures that an aborted conversion is not lost but takes part in the next arbitration round.

The trigger unit generates load events from the selected external (outside the ADC) trigger signals. For example, a timer unit can issue a request signal to synchronize conversions to PWM events.

Load events start a scan sequence and can be generated either via software or via the selected hardware triggers.

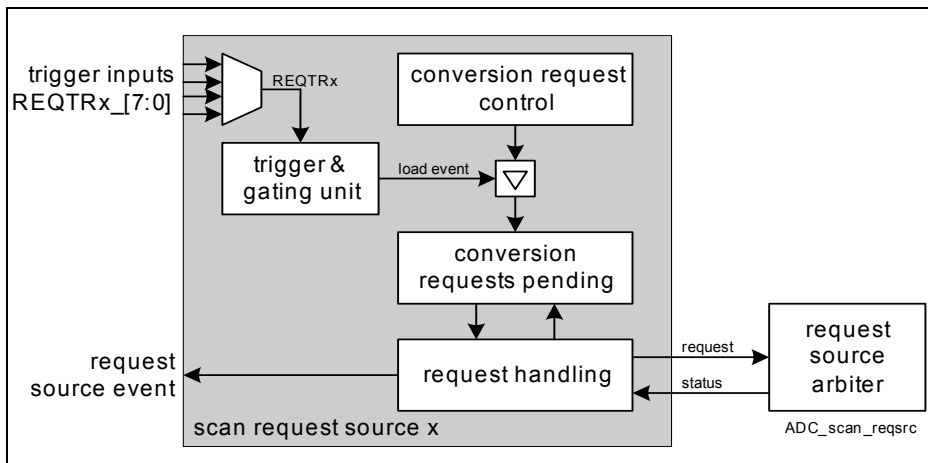


Figure 21-6 Scan Request Source

Scan Source Operation

Configure the scan request source by executing the following actions:

- Select the input channels for the sequence by programming **ADC_CRCR1**
- If hardware trigger is desired, select the appropriate trigger inputs by programming **ADC_ETRCR**. Enable the trigger by programming **ADC_CRMR1**.
- Define the load event operation (handling of pending bits, autoscan mode) by programming **ADC_CRMR1**.
A load event with bit LDM = 0 copies the content of **ADC_CRCR1** to **ADC_CRPR1** (overwrite mode). This starts a new scan sequence and aborts any pending conversions from a previous scan sequence.
A load event with bit LDM = 1 OR-combines the content of **ADC_CRCR1** to **ADC_CRPR1** (combine mode). This starts a scan sequence that includes pending conversions from a previous scan sequence.
- Enable the corresponding arbitration slot (1) to accept conversion requests from the channel scan source (see register **ADC_PRAR**).

Start a channel scan sequence by generating a load event:

- If a hardware trigger is selected and enabled, generate the configured transition at the selected input signal, e.g. from a timer or an input pin.
- Generate a software load event by setting **ADC_CRMR1.LDEV** = 1.
- Generate a load event by writing the scan pattern directly to the pending bits in **ADC_CRPR1**. The pattern is copied to **ADC_CRCR1** and a load event is generated automatically.
In this case, a scan sequence can be defined and started with a single data write action.

Note: If autoscan is enabled, a load event is generated automatically each time a request source event occurs when the scan sequence has finished. This permanently repeats the defined scan sequence (autoscan).

Stop or abort an ongoing scan sequence by executing the following actions:

- If external gating is enabled, switch the gating signal to the defined inactive level. This does not modify the conversion pending bits, but only prevents issuing conversion requests to the arbiter.
- Disable the corresponding arbitration slot (1) in the arbiter. This does not modify the contents of the conversion pending bits, but only prevents the arbiter from accepting requests from the request handling block.
- Disable the channel scan source by clearing bitfield **ENG**T = 0_B. Clear the pending request bits by setting bit **ADC_CRMR1.CLRPND** = 1.

Scan Request Source Events and Interrupts

A request source event of a scan source occurs if the last conversion of a scan sequence is finished (all pending bits = 0). A request source event interrupt can be generated based on a request source event according to the structure shown in [Figure 21-7](#). If a request source event is detected, it sets the corresponding indication flag in register [ADC_EVINFR](#). The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register [ADC_EVINCR](#).

The service request output SRx becomes activated each time the related request source event is detected (and enabled by CRMRx.ENS1).

The request source events and the result events share the same registers. The request source event is located at the bit position in register [ADC_EVINFR](#):

- Event 1: Request source event of the channel scan source 1 (in arbitration slot 1)

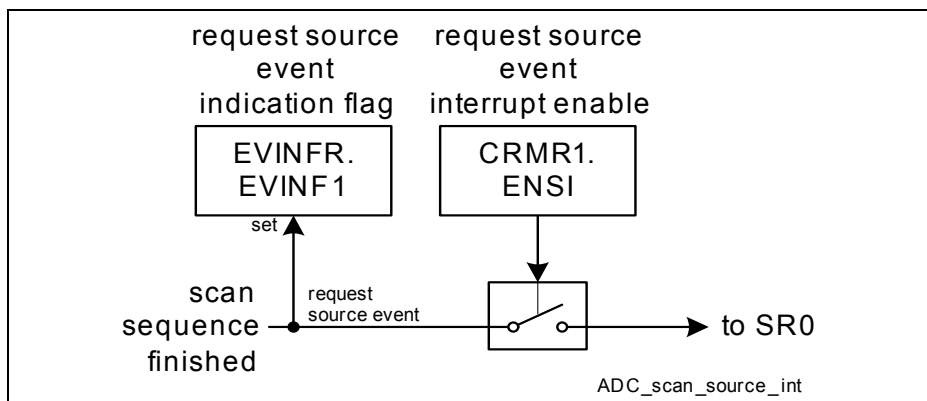


Figure 21-7 Interrupt Generation of a Scan Request Source

The conversion request mode registers contains bits used to set the request source in the desired mode.

ADC_CRMR1

Conversion Request Mode Register 1 (CC_H)

Reset Value: 00_H

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
0	LDEV	CLRPND	SCAN	ENSI	ENTR	0	0
r	w	w	rw	rw	rw	r	r

Analog to Digital Converter

Field	Bits	Type	Description
ENTR	2	rw	Enable External Trigger This bit enables the external trigger possibility. If enabled, the load event takes place if a rising edge is detected at the external trigger input REQTR. 0 _B The external trigger is disabled. 1 _B The external trigger is disabled.
ENSI	3	rw	Enable Source Interrupt This bit enables the request source interrupt. This interrupt can be generated when the last pending conversion is completed for this source (while PND = 0). 0 _B The source interrupt is disabled. 1 _B The source interrupt is enabled.
SCAN	4	rw	Autoscan Enable This bit enables the autoscan functionality. If enabled, the load event is automatically generated when a conversion (requested by this source) is completed and PND = 0. 0 _B The autoscan functionality is disabled. 1 _B The autoscan functionality is enabled.
CLRPND	5	w	Clear Pending Bits 0 _B No action 1 _B The bits in register CRPR1 are reset.
LDEV	6	w	Generate Load Event 0 _B No action 1 _B The load event is generated.
0	7,[1:0]	r	Reserved Returns 0 if read; should be written with 0.

Analog to Digital Converter

The Conversion Request 1 Control Register selects the channels to be converted by request source 1 (channel scan source). Its bits are used to update the pending register CRPR1, when the load event occurs.

Note: Writes to register CRPR1 also update CCR1 and generate a load event.

ADC_CRCR1

Conversion Request Control Register 1(CA_H)

Reset Value: 00_H

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CHx (x = 0 - 7)	x	rwh	<p>Channel Bit x</p> <p>Each bit corresponds to one analog channel, the channel number x is defined by the bit position in the register. The corresponding bit x in the conversion request pending register will be overwritten by this bit when the load event occurs.</p> <p>0_B The analog channel x will not be requested for conversion by the parallel request source.</p> <p>1_B The analog channel x will be requested for conversion by the parallel request source.</p> <p><i>Note: Bits 4-7 are only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i></p>

Analog to Digital Converter

The Conversion Request Pending Register indicates which channels of request source 1 (channel scan source) are requesting a conversion. Its bits are updated from pending register CRCR1, when the load event occurs.

Note: Writes to register CRPR1 also update CRCR1 and generate a load event.

ADC_CRPR1

Conversion Request Pending Register 1(CB_H)

Reset Value: 00_H

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
CHP7	CHP6	CHP5	CHP4	CHP3	CHP2	CHP1	CHP0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CHPx (x = 0 - 7)	x	rwh	Channel Pending Bit x Write view: A write to this address targets the bits in register CRCR1. Read view: Each bit corresponds to one analog channel; the channel number x is defined by the bit position in the register. The arbiter automatically resets (at start of conversion) or sets it again (at abort of conversion) for the corresponding analog channel. 0 _B The analog channel x is not requested for conversion by the parallel request source. 1 _B The analog channel x is requested for conversion by the parallel request source. <i>Note: Bits 4-7 are only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>
0	[7:4]	r	Reserved Returns 0 if read; should be written with 0.

Note: The bits that can be read from this register location are generally 'rh'. They cannot be modified directly by a write operation. A write operation modifies the bits in

Analog to Digital Converter

CRCR1 (that is why they are marked 'rwh') and leads to a load event one clock cycle later.

21.6.2 Queued Request Source Handling

A queued request source supports short conversion sequences of arbitrary channels (contrary to a scan request source with a fixed conversion order for the enabled channels). The programmed sequence is stored in a queue buffer (based on a FIFO mechanism). The requested channel numbers are entered via the queue input, while queue stage 0 defines the channel to be converted next.

A conversion request is only issued to the request source arbiter if a valid entry is stored in queue stage 0.

If the arbiter aborts a conversion triggered by a queued request source due to higher priority requests, the corresponding conversion parameters are automatically saved in the backup stage. This ensures that an aborted conversion is not lost but takes part in the next arbitration round (before stage 0).

The trigger and gating unit generates trigger events from the selected external (outside the ADC) trigger. For example, a timer unit can issue a request signal to synchronize conversions to PWM events.

Trigger events start a queued sequence and can be generated either via software or via the selected hardware triggers. The occurrence of a trigger event is indicated by bit QSRx.EV. This flag is cleared when the corresponding conversion is started or by writing to bit QMRx.CEV.

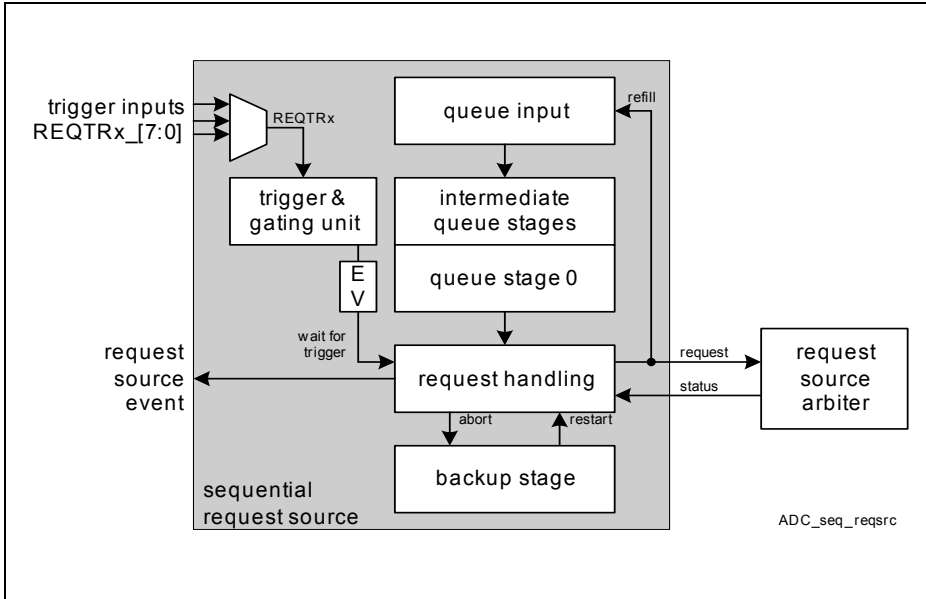


Figure 21-8 Queued Request Source

A sequence is defined by entering conversion requests into the queue input register ([ADC_QINR0](#)). Each entry selects the channel to be converted and can enable an external trigger, generation of an interrupt, and an automatic refill (i.e. keep this entry in the queue after conversion). The entries are stored in the queue buffer stages.

The content of stage 0 ([ADC_Q0R0](#)) selects the channel to be converted next. When the requested conversion is started, the contents of this queue stage is invalidated and copied to the backup stage. Then the next queue entry can be handled (if available).

Note: The contents of the queue stages cannot be modified directly, but only by writing to the queue input or by flushing the queue.

If all queue entries have automatic refill selected, the defined conversion sequence can be repeated without re-programming.

Properties of the Queued Request Source

The ADC kernels of the XC82x provide one queued request source with buffer size:

- Queued request source 0 provides 4 buffer stages and can handle sequences of up to 4 input channel entries. It supports short application-specific conversion sequences, especially for timing-critical sequences containing also multiple conversions of the same channel.

Queued Source Operation

Configure the queued request source by executing the following actions:

- Define the sequence by writing the entries to the queue input **ADC_QINR0**. Initialize the complete sequence before enabling the request source, because with enabled refill feature, software writes to QINRx are not allowed.
- If hardware trigger is desired, select the appropriate trigger inputs by programming **ADC_ETRCR**.
Enable the trigger by programming bitfield ENGT in register **ADC_QMR0**.
- Enable the corresponding arbitration slot (0) to accept conversion requests from the queued source (see register **ADC_PRAR**).

Start a queued sequence by generating a trigger event:

- If a hardware trigger is selected and enabled, generate the configured transition at the selected input signal, e.g. from a timer or an input pin.
- Generate a software trigger event by setting QMRx.TREV = 1.
- Write a new entry to the queue input of an empty queue. This leads to a (new) valid queue entry that is forwarded to queue stage 0 and starts a conversion request (if enabled by QMRx.ENG T and without waiting for an external trigger).

Note: If the refill mechanism is activated, a processed entry is automatically reloaded into the queue. This permanently repeats the respective sequence (autoscan). In this case, do not write to the queue input while the queued source is running. Write operations to a completely filled queue are ignored.

Stop or abort an ongoing queued sequence by executing the following actions:

- Disable the corresponding arbitration slot (0) in the arbiter. This does not modify the queue entries, but only prevents the arbiter from accepting requests from the request handling block.
- Disable the queued source by clearing bitfield ENGT = 0_B.
 - Invalidate the next pending queue entry by setting bit QMRx.CLRV = 1.
If the backup stage contains a valid entry, this one is invalidated, otherwise stage 0 is invalidated.
 - Remove all entries from the queue by setting bit QMRx.FLUSH = 1.

Queue Request Source Events and Interrupts

A request source event of a queued source occurs when a conversion is finished. A request source event interrupt can be generated based on a request source event according to the structure shown in [Figure 21-9](#). If a request source event is detected, it sets the corresponding indication flag in register [ADC_EVINFR](#). The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register [ADC_EVINCR](#).

The interrupt enable bit is taken from stage 0 for a normal sequential conversion, or from the backup stage for a repeated conversion after an abort.

The service request output line SRx becomes activated each time the related request source event is detected (and enabled by Q0Rx.ENS1, or QBURx.ENS1 respectively).

The request source events and the result events share the same registers. The request source event is located at the bit position in register [ADC_EVINFR](#):

- Event 0: Request source event of queued source 0 (in arbitration slot 0)

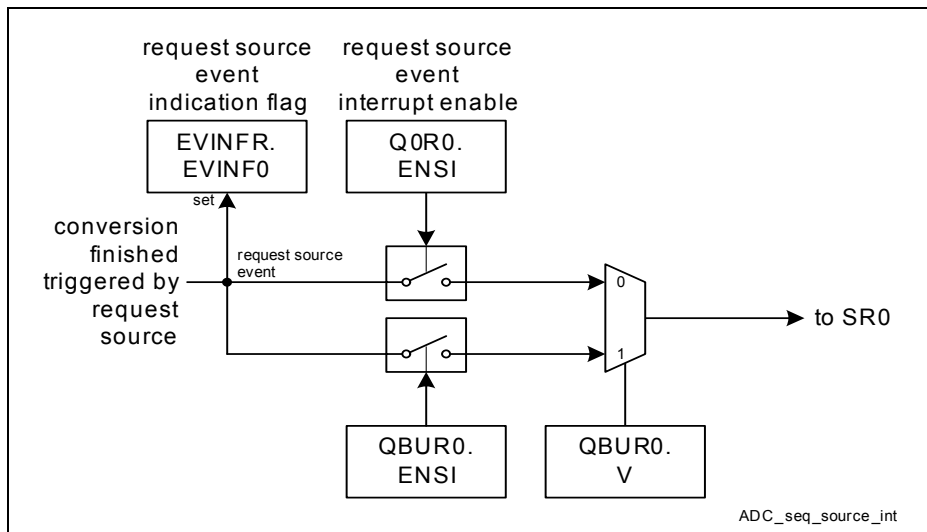


Figure 21-9 Interrupt Generation of a Queued Request Source

Analog to Digital Converter

The Queue Mode Register configures the operating mode of a queued request source.

ADC_QMR0

Queue Mode Register

(CD_H)

Reset Value: 00_H

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
CEV	TREV	FLUSH	CLRV	0	ENTR	0	ENGT
w	w	w	w	r	rw	r	rw

Field	Bits	Type	Description
ENGT	0	rw	Enable Gate This bit enables the gating functionality for the request source. 0 _B The gating line is permanently 0. The source is switched off. 1 _B The gating line is permanently 1. The source is switched on.
ENTR	2	rw	Enable External Trigger This bit enables the external trigger possibility. If enabled, bit EV is set if a rising edge is detected at the external trigger input REQTR when at least one V bit is set in register Q0R0 or QBUR0. 0 _B The external trigger is disabled. 1 _B The external trigger is enabled.
CLRV	4	w	Clear V Bits 0 _B No action 1 _B The bit V in register Q0R0 or QBUR0 is reset. If QBUR0.V = 1, then QBUR0.V is reset. If QBUR0.V = 0, then Q0R0.V is reset.
FLUSH	5	w	Flush Queue 0 _B No action 1 _B All bits V in the queue registers and bit EV are reset. The queue contains no more valid entry.
TREV	6	w	Trigger Event 0 _B No action 1 _B A trigger event is generated by software. If the source waits for a trigger event, a conversion request is started.

Analog to Digital Converter

Field	Bits	Type	Description
CEV	7	w	Clear Event Bit 0_B No action 1_B Bit EV is cleared.
0	1, 3	r	Reserved Returns 0 if read; should be written with 0.

*Note: Before SW modifies the queue content by QMR.CLRV or QMR.FLUSH, all HW actions related to this queue have to be finished. Therefore, the arbitration slot has to be disabled and SW has to wait for at least two arbitration rounds (to be sure that this request source can no longer be an arbitration winner). Then, it has to check **ADC_GLOBCTR.BUSY** to be sure that a conversion triggered by this request source is no longer running. Then SW can read QBURx and Q0Rx and can start modification of the queue content.*

Analog to Digital Converter

The Queue Status Register indicates the current status of the queued source. The filling level and the empty information refer to the queue intermediate stages (if available) and to the queue register 0. An aborted conversion stored in the backup stage is not indicated by these bits (therefore, see QBURx.V).

ADC_QSR0

Queue Status Register

(CE_H)

Reset Value: 20_H

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
0	EMPTY	EV	0			FILL	
r	rh	rh	r			rh	

Field	Bits	Type	Description
FILL	[1:0]	rh	Filling Level This bit field indicates how many entries are valid in the sequential-sourced queue. It is incremented each time a new entry is written to QINR0, decremented each time a requested conversion has been finished. A new entry is ignored if the filling level has reached its maximum value. If EMPTY bit = 1, there are no valid entries in the queue. 00 _B If EMPTY bit = 0, there is 1 valid entry in the queue. 01 _B If EMPTY bit = 0, there are 2 valid entries in the queue. 10 _B If EMPTY bit = 0, there is 3 valid entry in the queue. 11 _B If EMPTY bit = 0, there are 4 valid entries in the queue.
EV	4	rh	Event Detected This bit indicates that an event has been detected while V = 1. Once set, this bit is reset automatically when the requested conversion is started. 0 _B An event has not been detected. 1 _B An event has been detected.

Analog to Digital Converter

Field	Bits	Type	Description
EMPTY	5	rh	Queue Empty This bit indicates if the sequential source contains valid entries. A new entry is ignored if the queue is filled (EMPTY = 0). 0 _B The queue is filled with 'FILL+1' valid entries in the queue. 1 _B The queue is empty, no valid entries are present in the queue.
0	[3:2], [7:6]	r	Reserved Returns 0 if read; should be written with 0.

Analog to Digital Converter

The Queue Input Register is the entry point for conversion requests of a queued request source.

ADC_QINR0

Queue Input Register 0

(D2_H)

Reset Value: 00_H

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
EXTR	ENSI	RF	0				REQCHNR
w	w	w	r			w	

Field	Bits	Type	Description
REQCHNR	[2:0]	w	Request Channel Number This bit field defines the requested channel number. <i>Note: Bit 2 is only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>
RF	5	w	Refill This bit defines the refill functionality.
ENSI	6	w	Enable Source Interrupt This bit defines the source interrupt functionality.
EXTR	7	w	External Trigger This bit defines the external trigger functionality.
0	[4:3]	r	Reserved Returns 0 if read; should be written with 0.

Analog to Digital Converter

The queue registers 0 monitor the status of the pending request (queue stage 0).

ADC_Q0R0

Queue 0 Register 0

(CF_H)

Reset Value: 00_H

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
EXTR	ENSI	RF	V	0	REQCHNR		
rh	rh	rh	rh	r	rh		

Field	Bits	Type	Description
REQCHNR	[2:0]	rh	Request Channel Number This bit field indicates the channel number that will be or is currently requested. <i>Note: Bit 2 is only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>
V	4	rh	Request Channel Number Valid This bit indicates if the data in REQCHNR, RF, ENSI and EXTR is valid. Bit V is set when a valid entry is written to the queue input register QINR0 (or by an update by intermediate queue registers). 0 _B The data is not valid. 1 _B The data is valid.
RF	5	rh	Refill This bit indicates if the pending request is discarded after being executed (conversion start) or if it is automatically refilled in the top position of the request queue. 0 _B The request is discarded after conversion start. 1 _B The request is refilled in the queue after conversion start.

Analog to Digital Converter

Field	Bits	Type	Description
ENSI	6	rh	Enable Source Interrupt This bit indicates if a source interrupt will be generated when the conversion is completed. The interrupt trigger becomes activated if the conversion requested by the source has been completed and ENSI = 1. 0 _B The source interrupt generation is disabled. 1 _B The source interrupt generation is enabled.
EXTR	7	rh	External Trigger This bit defines if the conversion request is sensitive to an external trigger event. The event flag (bit EV) indicates if an external event has taken place and a conversion can be requested. 0 _B Bit EV is not used to start conversion request. 1 _B Bit EV is used to start conversion request.
0	3	r	Reserved Returns 0 if read; should be written with 0.

Analog to Digital Converter

The Queue Backup Registers monitor the status of an aborted queued request.

ADC_QBUR0

Queue Backup Register 0

(D2_H)

Reset Value: 00_H

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
EXTR	ENSI	RF	V	0	REQCHNR		
rh	rh	rh	rh	r	rh		

Field	Bits	Type	Description
REQCHNR	[2:0]	rh	Request Channel Number This bit field is updated by bit field Q0R0.REQCHNR when the conversion requested by Q0R0 is started. <i>Note: Bit 2 is only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>
V	4	rh	Request Channel Number Valid This bit indicates if the data in REQCHNR, RF, ENSI, and EXTR is valid. Bit V is set if a running conversion is aborted. It is reset when the conversion is started. 0 _B The backup register does not contain valid data, because the conversion described by this data has not been aborted. 1 _B The data is valid. The aborted conversion is requested before taking into account what is requested by Q0R0.
RF	5	rh	Refill This bit is updated by bit Q0R0.RF when the conversion requested by Q0R0 is started.
ENSI	6	rh	Enable Source Interrupt This bit is updated by bit Q0R0.ENSI when the conversion requested by Q0R0 is started.
EXTR	7	rh	External Trigger This bit is updated by bit Q0R0.EXTR when the conversion requested by Q0R0 is started.
0	3	r	Reserved Returns 0 if read; should be written with 0.

Analog to Digital Converter

Note: Registers QBURx share addresses with registers QINRx.

Read operations return the status bits from register QBURx. Write operations target the control bits in register QINRx.

21.6.3 Hardware Trigger Selection

Each request source can be activated either by software or by a hardware trigger signal. The hardware triggers can be derived from several module signals or port inputs.

The external trigger control register contains bits to select the external trigger input signal source.

ADC_ETRCR

External Trigger Control Register (D3_H)

Reset Value: 00_H

RMAP: 0, PAGE: 4

7	6	5	4	3	2	1	0
0		ETRSEL1			ETRSEL0		
r		rw			rw		

Field	Bits	Type	Description
ETRSEL0, ETRSEL1	[2:0], [5:3]	rw	External Trigger Selection for Request Source x This bit field defines which external trigger input signal is selected. 000 _B The trigger input REQTRxA is selected. 001 _B The trigger input REQTRxB is selected. 010 _B The trigger input REQTRxC is selected. 011 _B The trigger input REQTRxD is selected. 100 _B The trigger input REQTRxE is selected. 101 _B The trigger input REQTRxF is selected. 110 _B The trigger input REQTRxG is selected. 111 _B The trigger input REQTRxH is selected.
0	[7:6]	r	Reserved Returns 0 if read; should be written with 0.

21.7 Request Source Arbitration

The request source arbiter regularly polls the request sources, one after the other, for pending conversion requests. Each request source is assigned to a certain time slot within an arbitration round, called arbitration slot.

The priority of each request source is user-configurable via register **ADC_PRAR**, so the arbiter can select the next channel to be converted, in the case of concurrent requests from multiple sources, according to the application requirements.

A disabled or unused arbitration slot is considered empty and does not take part in the arbitration. After reset, all slots are disabled and must be enabled (register **ADC_PRAR**) to take part in the arbitration process.

Figure 21-10 summarizes the arbitration sequence. An arbitration round consists of one arbitration slot for each available request source. The synchronization source is always evaluated in the last slot and has a higher priority than all other sources. Additional arbitration slots can be inserted to adjust the timing to other products (not required for the XC82x). At the end of each arbitration round, the arbiter has determined the highest priority conversion request.

If a conversion is started in an arbitration round, this arbitration round does not deliver an arbitration winner.

In the XC82x, the following request sources are available:

- Arbitration slot 0: **4-stage sequential source**, 4-stage sequences in arbitrary order
- Arbitration slot 1: **4/8-channel scan source**, sequences in defined order
- Last arbitration slot: **Synchronization source**, synchronized conversion requests from another ADC kernel (always handled with the highest priority in a synchronization slave kernel).

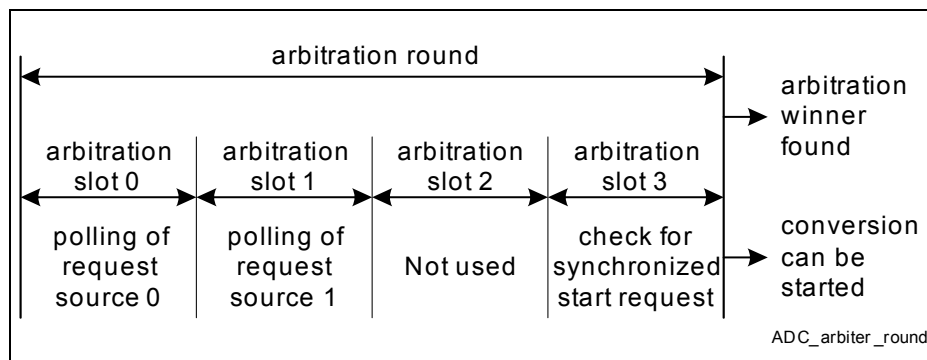


Figure 21-10 Arbitration Round

21.7.1 Arbiter Timing

The timing of the arbiter (i.e. of an arbitration round) is determined by the number of arbitration slots within an arbitration round and by the duration of an arbitration slot.

An arbitration round consist of 4arbitration slots

The duration of an arbitration slot is configurable $t_{\text{Slot}} = f_{\text{ADC}}$.

The duration of an arbitration round, therefore, is $t_{\text{ARB}} = N \times t_{\text{Slot}}$ (N = number of slots).

The period of the arbitration round introduces a timing granularity to detect an incoming conversion request signal and the earliest point to start the related conversion. This granularity can introduce a jitter of maximum one arbitration round. The jitter can be reduced by minimizing the period of an arbitration round.

To achieve a reproducible reaction time (constant delay without jitter) between the trigger event of a conversion request (e.g. by a timer unit or due to an external event) and the start of the related conversion, mainly the following two options exist. For both options, the converter has to be idle and other conversion requests must not be pending for at least one arbiter round before the trigger event occurs:

- If bit **ADC_PRAR.ARB** = 0, the **arbiter runs permanently**.
The trigger for the conversion triggers has to be generated synchronously to the arbiter timing. Incoming triggers should have exactly n-times the granularity of the arbiter ($n = 1, 2, 3, \dots$). In order to allow some flexibility, the duration of an arbitration slot can be programmed in cycles of f_{ADC} .
- If bit **ADC_PRAR.ARB** = 1, the **arbiter stops after an arbitration round** when no conversion request have been found pending any more. The arbiter is started again if at least one enabled request source indicates a pending conversion request. The trigger of a conversion request does need not to be synchronous to the arbiter timing.

21.7.2 Request Source Priority and Conversion Start Mode

Each request source has a configurable priority, so the arbiter can resolve concurrent conversion requests from different sources. The request with the highest priority is selected for conversion. These priorities can be adapted to the requirements of a given application (see register [ADC_PRAR](#)).

The **Conversion Start Mode** determines the handling of the conversion request that has won the arbitration.

The Priority and Arbitration Register defines the request source priority, the conversion start mode for each request source and enables/disables the arbitration slots to control whether or not conversion requests are considered.

Note: Only change priority and conversion start mode settings of a request source while this request source is disabled, and a currently running conversion requested by this source is finished.

ADC_PRAR

Priority and Arbitration Register

(CC_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
ASEN1	ASEN0	0	ARBM	CSM1	PRI01	CSM0	PRI00
rw	rw	r	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PRI00	0	rw	Priority of Request Source 0 This bit defines the priority of the sequential request source 0. 0 _B Low priority 1 _B High priority
CSM0	1	rw	Conversion Start Mode of Request Source 0 This bit defines the conversion start mode of the sequential request source 0. 0 _B The wait-for-start mode is selected. 1 _B The cancel-inject-repeat mode is selected.
PRI01	2	rw	Priority of Request Source 1 This bit defines the priority of the parallel request source 1. 0 _B Low priority 1 _B High priority

Analog to Digital Converter

Field	Bits	Type	Description
CSM1	3	rw	Conversion Start Mode of Request Source 1 This bit defines the conversion start mode of the parallel request source 1. 0_B The wait-for-start mode is selected. 1_B The cancel-inject-repeat mode is selected.
ARBM	4	rw	Arbitration Mode This bit defines which arbitration mode is selected. 0_B Permanent arbitration (default). 1_B Arbitration started by pending conversion request
ASEN0, ASEN1	6, 7	rw	Arbitration Slot x Enable Each bit enables an arbitration slot of the arbiter round. ASEN0 enables arbitration slot 0, ASEN1 enables slot 1. If an arbitration slot is disabled, a pending conversion request of a request source connected to this slot is not taken into account for arbitration. 0_B The corresponding arbitration slot is disabled. 1_B The corresponding arbitration slot is enabled.
0	5	r	Reserved Returns 0 if read; should be written with 0.

Note: If the arbiter shall not be running continuously ($ARBM = 1$), no conversion request of the request source for arbitration slot x must be active. Clear conversion requests of the related request source before disabling an arbitration slot.

Conversion Start Mode

When the arbiter has selected the request to be converted next, the handling of this channel depends on the current activity of the converter:

- Converter is currently idle: the conversion of the arbitration winner is started immediately.
- Current conversion has same or higher priority: the current conversion is completed, the conversion of the arbitration winner is started after that.
- Current conversion has lower priority: the action is user-configurable:
 - **Wait-for-start mode:** the current conversion is completed, the conversion of the arbitration winner is started after that. This mode provides maximum throughput, but can produce a jitter for the higher priority conversion.

Example in [Figure 21-11](#):

Conversion A is requested (t1) and started (t2). Conversion B is then requested (t3), but started only after completion of conversion A (t4).

- **Cancel-inject-repeat mode:** the current conversion is aborted, the conversion of the arbitration winner is started after the abortion ($1 \dots 3 \cdot f_{\text{ADCI}}$ cycles).

The aborted conversion request is restored in the corresponding request source and takes part again in the next arbitration round. This mode provides minimum jitter for the higher priority conversions, but reduces the overall throughput.

Example in [Figure 21-11](#):

Conversion A is requested (t6) and started (t7). Conversion B is then requested (t8) and started (t9), while conversion A is aborted but requested again. When conversion B is complete (t10), conversion A is restarted.

Exception: If both requests target the same result register with wait-for-read mode active (see [Section 21.10](#)), the current conversion cannot be aborted.

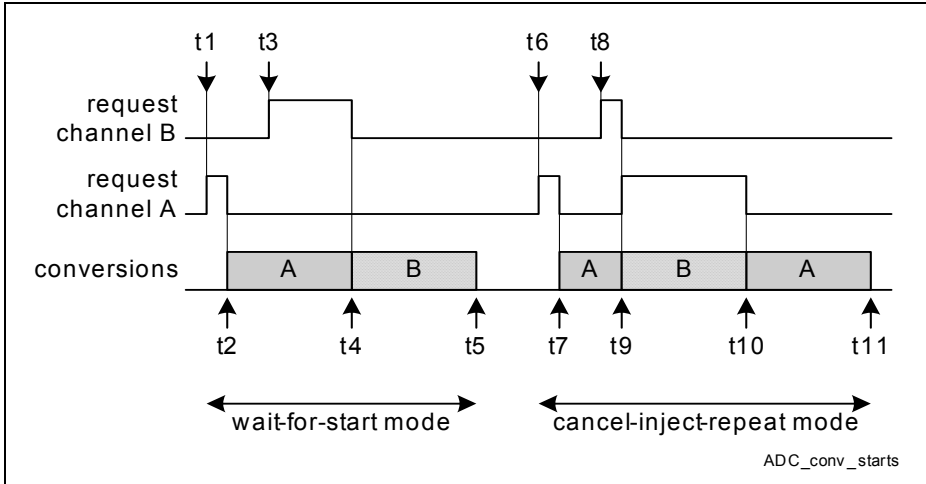


Figure 21-11 Conversion Start Modes

The conversion start mode can be individually programmed for each request source by bits in register **ADC_PRAR** and is applied to all channels requested by the source. In this example, channel A is issued by a request source with a lower priority than the request source requesting the conversion of channel B.

21.8 Analog Input Channel Configuration

For each analog input channel, a number of parameters can be configured that control the conversion of this channel. After a channel has won the arbitration, its parameters are applied to the converter. The channel control registers (CHCTR_x on [Page 21-48](#)) define the following parameters:

- **Conversion parameters:** The input class defines sample time and data width. All channels is using the same input class.
- **Reference selection:** Three types of internal reference are available for selection ranging from internal 1.2V and V_{DDP} 3.3/5V. The reference selection determines the conversion conversion mode of the ADC, i.e single ended or differential mode. See [Section 21.8.1](#)
Please note that low reference voltages lead to small granularity. As a consequence the resulting TUE increases due to noise effects.
- **Result target:** The conversion result can be stored in one of 4 result registers.
- **Channel event handling:** Channel events can be restricted to results inside or outside a defined area of values (limit checking).

In addition to the general channel control, the ADC kernel supports a mechanism (named alias feature, see [Section 21.8.4](#)) to redirect a conversion request to another channel number.

21.8.1 Reference Selection

The conversion result of the ADC is always reference to a reference voltage. The maximum digital result value (full scale) is obtained if the analog input voltage equals the reference voltage. In order to support more than one measurement range with full scale digital representation, the user can select between three conversion modes which are:

- Single ended measurement with ADC reference connected internally to V_{DDP} and V_{SSP}. See [Figure 21-12](#)
- Differential like measurement with ADC reference connected to internal 1.2V voltage reference and ground reference taken from CH0. See [Figure 21-13](#)
- Single ended measurement with ADC reference connected to internal 1.2V voltage reference and V_{SSP}. See [Figure 21-14](#)

In single ended ADC conversion with reference to V_{DDP} and V_{SSP} voltage, the ADC reference voltage is internally connected to V_{DDP} and V_{SSP} voltage. See [Figure 21-12](#)

Analog to Digital Converter

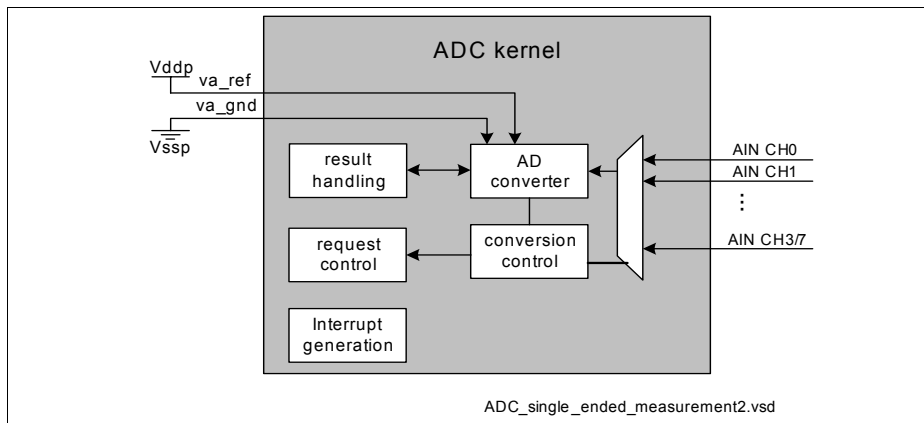


Figure 21-12 Single ended measurement with Vddp, Vssp

In differential like ADC conversion with internal 1.2V voltage reference and ground reference taken from CH0. CH0 cannot be used for ADC measurement. CH0 is used as ground reference for ADC which may be connected externally to increase measurement accuracy. See [Figure 21-13](#). The digital features of CH0 maybe reused for other channels with the alias function. See [Section 21.8.4](#). Please note that a minimum time of 50usec is needed between conversions, when this mode is used.

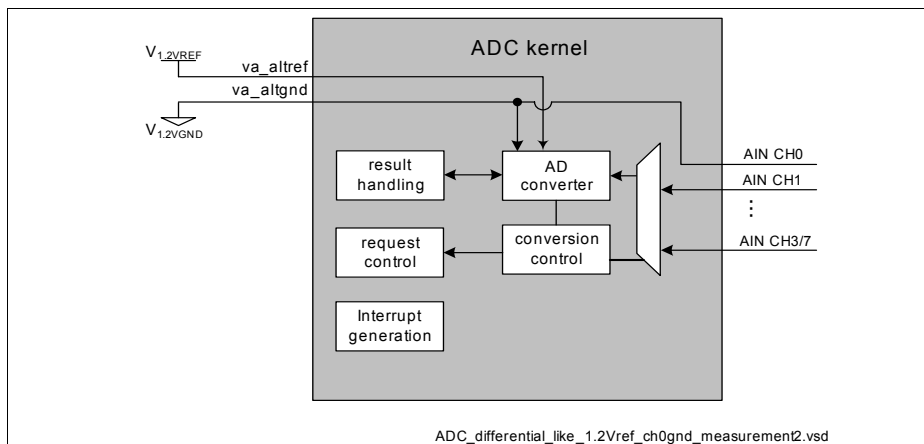


Figure 21-13 Differential like measurement with internal 1.2V voltage reference, and CH0 gnd.

In single ended ADC conversion with 1.2V voltage reference and Vssp, CH0 can be used for ADC measurement. See [Figure 21-14](#)

Analog to Digital Converter

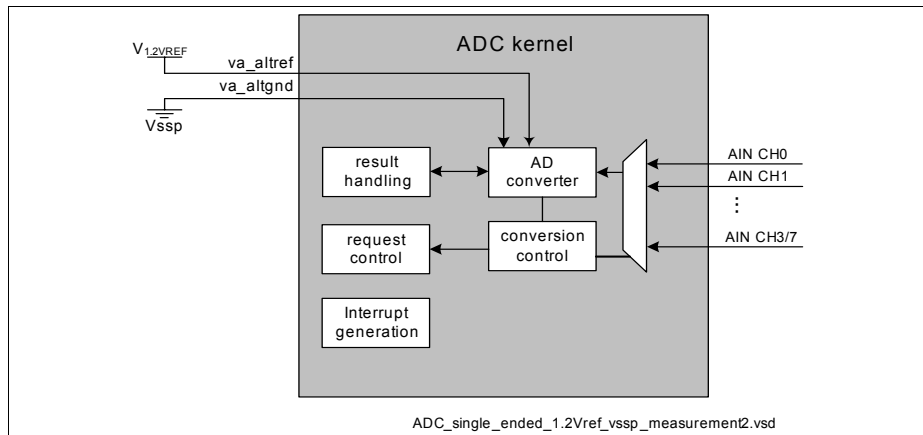


Figure 21-14 Single ended measurement with internal 1.2V voltage reference and Vssp.

The reference selection for each input channel is selected by programming `ADC_CHCTRx.REFSEL` bits. Please note that a minimum time of 50 μ sec is needed between conversions, when this mode is used.

21.8.2 Channel Parameters

Each analog input channel is configured by its associated channel control register. The sample time and the result width are selected via an input class.

The Channel Control Registers select the control parameters for each input channel, it contain bits to select the targetted result register, selection of internal reference voltages, controls the limit check mechanism and boundary flags.

ADC_CHCTR_x (x = 0 - 2)

Channel x Control Register

(CA_H + x * 1)

Reset Value: 00_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
BFEN		LCC		REFSEL		RESRSEL	
rw		rw		rw		rw	

Field	Bits	Type	Description
RESRSEL	[1:0]	rw	Result Register Selection This bit field defines which result register will be the target of a conversion of this channel. 00 _B The result register 0 is selected. 01 _B The result register 1 is selected. 10 _B The result register 2 is selected. 11 _B The result register 3 is selected.
REFSEL	[3:2]	rw	Reference Input Selection This bit field defines the reference source for this channel. 00 _B Analog to digital conversion is done with reference to V _{DDP} , V _{SSP} (See Figure 21-12). 01 _B Analog to digital conversion is done with internal 1.2V and CH0 as the ground reference (See Figure 21-13) ¹⁾ . 10 _B Analog to digital conversion is done with internal 1.2V and Vssp (See Figure 21-14) ¹⁾ . 11 _B Reserved; do not use this combination

Analog to Digital Converter

Field	Bits	Type	Description
LCC	[6:4]	rw	Limit Check Control This bit field defines the behavior of the limit checking mechanism. See Section 21.8.3 000 _B Never 001 _B Result outside area I 010 _B Result outside area II 011 _B Result outside area III 100 _B Always (boundaries disregarded) 101 _B Result within area I 110 _B Result within area II 111 _B Result within area III
BFEN	7	rw	Boundary Flags Enable This bit is the gating control for the boundary flags ADC_BF<2:0> signal. 0 _B Boundary flag signal is disabled for the corresponding channel. 1 _B Boundary flag signal is enabled for the corresponding channel.

1) A minimum of 50µsec is needed between conversions when using this mode.

ADC_CHCTR_x (x = 3 - 5)

Channel x Control Register

(CA_H + x * 1)

Reset Value: 00_H

ADC_CHCTR_x (x = 6 - 7)

Channel x Control Register

(CC_H + x * 1)

Reset Value: 00_H

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	LCC			REFSEL		RESRSEL	
r	rw			rw		rw	

Note: ADC_CHCTR₄ - 7 are only for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.

Analog to Digital Converter

Field	Bits	Type	Description
RESRSEL	[1:0]	rw	Result Register Selection This bit field defines which result register will be the target of a conversion of this channel. 00 _B The result register 0 is selected. 01 _B The result register 1 is selected. 10 _B The result register 2 is selected. 11 _B The result register 3 is selected.
REFSEL	[3:2]	rw	Reference Input Selection This bit field defines the reference source for this channel. 00 _B Analog to digital conversion is done with reference to V _{DDP} , V _{SSP} (See Figure 21-12). 01 _B Analog to digital conversion is done with internal 1.2V and CH0 as the ground reference (See Figure 21-13) ¹⁾ . 10 _B Analog to digital conversion is done with internal 1.2V and Vssp (See Figure 21-14) ¹⁾ . 11 _B Reserved; do not use this combination
LCC	[6:4]	rw	Limit Check Control This bit field defines the behavior of the limit checking mechanism. See Section 21.8.3 000 _B Never 001 _B Result outside area I 010 _B Result outside area II 011 _B Result outside area III 100 _B Always (boundaries disregarded) 101 _B Result within area I 110 _B Result within area II 111 _B Result within area III
0	7	r	Reserved Returns 0 if read; should be written with 0.

1) A minimum of 50usec is needed between conversions when using this mode.

Analog to Digital Converter

An input class defines the length of the sample phase and the resolution of the conversion.

The default settings select the minimum sample phase length of $2f_{\text{ADCl}}$ cycles.

The Input Class Registers select the sample time and the resolution for each input class.

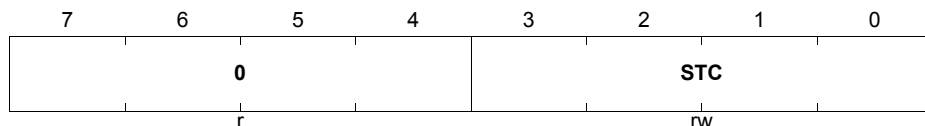
ADC_INPCR0

Input Class 0 Register

(CE_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0



Field	Bits	Type	Description
STC	[3:0]	rw	Sample Time Control This bit field defines the additional length of the sample time, given in terms of f_{ADCl} clock cycles. A sample time of 2 analog clock cycles is extended by the programmed value.
0	[7:4]	r	Reserved Returns 0 if read; should be written with 0.

21.8.3 Limit Checking

The limit checking mechanism automatically compares each conversion result to two boundary values (boundary A and boundary B). For each channel, the user can select these boundaries from a set of 2 programmable values (**ADC_LCBR0** to **ADC_LCBR1**). A channel event is then generated depending on the two comparisons. The conditions are selected via bitfield LCC in the respective channel control register.

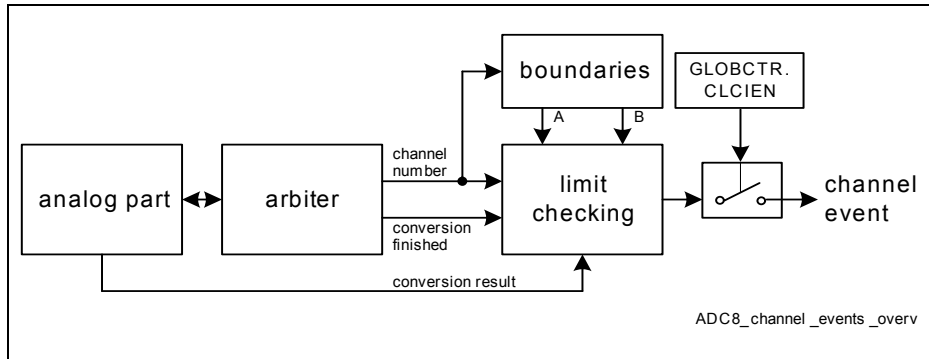


Figure 21-15 Channel Event Generation

The two selectable boundaries split the conversion result range into three areas:

- Area I: Conversion result below or equal to both boundaries.
- Area II: Conversion result above one boundary and below/equal to other boundary.
- Area III: Conversion result above both boundaries.

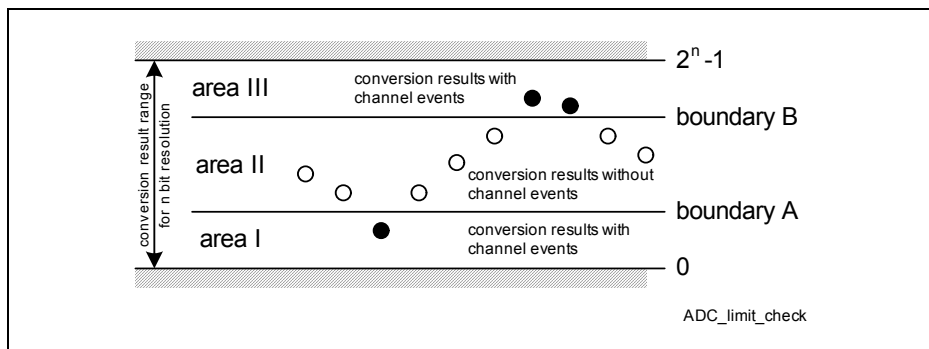


Figure 21-16 Limit Checking

The shown example for limit checking generates channel events only if the conversion results are outside the normal operating range defined by area II (LCC = 010_B).

Analog to Digital Converter

If only two areas are required, use the same boundary register for boundary A and boundary B. In this case, area II is empty and two result ranges are available. Avoid $LCC = x10_B$ in this case.

Typical applications for limit checking are monitoring tasks (temperature, pressure, current, etc.) where the real value of a result is less important than its range. As long as the measured values are within their defined valid range, no CPU action is required. A channel event should be generated only if the conversion result is outside the valid range to indicate a critical condition (over-temperature, loss of pressure, etc.).

The CPU load is minimized if the conversions of the analog input signals to be monitored are part of an auto-scan sequence autonomously triggered on a regular time base. Under normal conditions the CPU load here is zero.

Note: In the case of an over-current protection, the channel event can be used to disable PWM generation to reduce the current (in the XC82x, an interrupt output line of the ADC module is connected to a corresponding input of the CCU6x units to allow fast reactions without CPU intervention).

Boundary Flag Control

The limit checking mechanism can be configured to automatically control the boundary flags. A boundary flag will be set when the conversions result is within area III, and will be cleared when the conversion result is within area I.

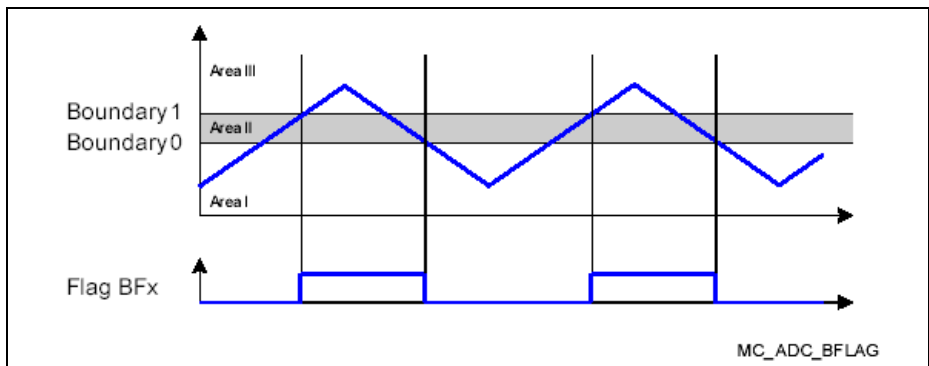


Figure 21-17 Boundary Flag Control

The difference between the two boundary values defines a hysteresis for setting/clearing the boundary flags.

Using this feature on three channels that monitor linear hall elements can produce signals to feed the three hall position inputs of a CCU6x unit.

Analog to Digital Converter

The Limit Check Boundary Registers define compare values (boundaries) for the limit checking unit.

ADC_LCBR0

Limit Check Boundary Register 0 (CD_H)

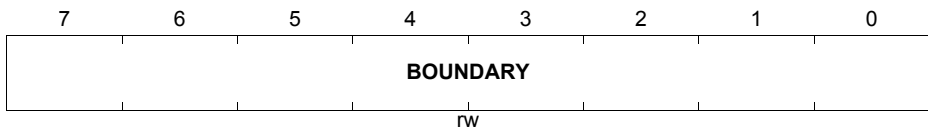
Reset Value: 70_H

ADC_LCBR1

Limit Check Boundary Register 1 (CF_H)

Reset Value: B0_H

RMAP: 0, PAGE: 0



Field	Bits	Type	Description
BOUNDARY	[7:0]	rw	Boundary for Limit Checking This value is compared to the actual conversion result.

21.8.4 Alias Feature

The alias feature re-directs conversion requests for channels CH0 to other channel numbers. This means that CH0 are converted with the channel parameters of the referenced channel instead of with their own. The re-direction feature serves several purposes:

- The same signal can be measured twice and the two results (original and re-directed) can be stored in separate result registers. This allows triggering both conversions quickly one after the other while data loss is avoided, independent from the CPU interrupt latency.
The sensor signal is connected to only one input (instead of two). This can save input pins in low-cost applications and reduces the input leakage to be considered in the error calculation.
- Even if the analog input CH0 is used as alternate gnd (see [Figure 21-18](#)), the internal trigger and data handling features for channel CH0 can be used.
- The channel settings for both conversions (of the same signal) can be different in terms of boundary values, interrupts, etc.
- If a queued conversion request source has been set up, a conversion request for channels CH0 can be easily re-directed to other input channels without flushing the queue.

[Figure 21-18](#) shows an example where the sensor signal is connected to one input channel (CHx) but two conversions are triggered (for CHx and CH0). The alias feature

Analog to Digital Converter

re-directs the conversion request for CH0 to CHx, but taking into account the settings for CH0. Although the same analog input (CHx) has been measured, the conversion results can be stored and retrieved from result registers RESRx (conversion triggered for CHx) and RESR0 (conversion triggered for CH0). Additionally, different interrupts or limit boundaries can be selected, enabled or disabled.

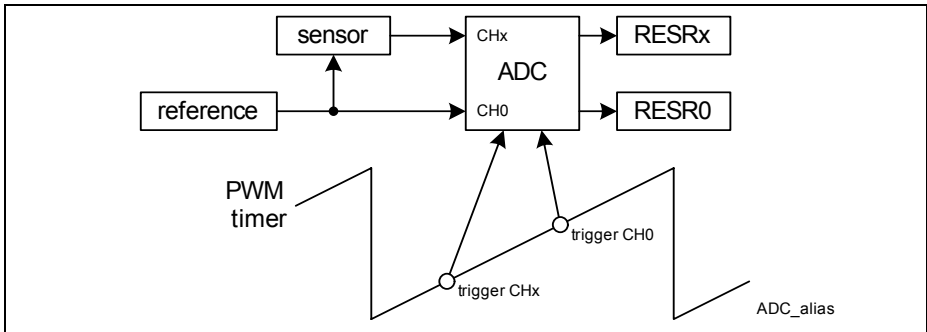


Figure 21-18 Alias Feature

In typical low-cost AC-drive applications, only one common current sensor is used to determine the phase currents. Depending on the applied PWM pattern, the measured value has different meanings and the sample points have to be precisely located in the PWM period.

Analog to Digital Converter

The Alias Register specifies replacement channel numbers for CH0, i.e. CH0 will use the respective channel numbers when requested.

The programmed alias channel number controls the analog input multiplexer (of the converter). The original channel number controls all other internal actions and the synchronization request.

ADC_ALR0

Alias Register 0

(CF_H)

Reset Value: 00_H

RMAP: 0, PAGE: 4

7	6	5	4	3	2	1	0
0					ALIAS0		
r					rw		

Field	Bits	Type	Description
ALIAS0	[2:0]	rw	Alias Value for CH0 Conversion Requests The channel indicated in this bit field is converted instead of channel CH0. The conversion is done with the settings defined for channel CH0. The default for this is CH0. <i>Note: Bit 2 is only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>
0	[7:3]	r	Reserved Returns 0 if read; should be written with 0.

21.8.5 Out of Range Comparator

The out of range comparator mechanism is build into every ADC channels and detects voltages higher or lower than V_{ddp} . Before using the out of range comparator, it has to be enabled by setting the bits at **ADC_ENORC**.ENORCx and configuring it to detect voltage higher or lower than V_{ddp} by setting bits **ADC_CORC**.CNF_x (See [Figure 21-19](#)).

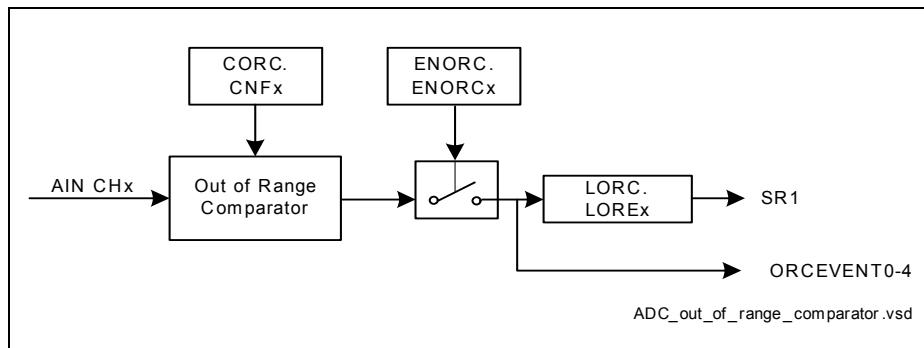


Figure 21-19 Out of range comparator

When a voltage out of range event occurs, the event is latched into **ADC_LORC**.LORE_x, sets an interrupt request through ADC_SR1 and triggers other modules through internal connections with ORCEVENT0-4. Triggering of other modules is only made available on ADC_CH0-4.

Analog to Digital Converter

The bit fields in these registers enables the out of range comparator in the ADC channel.

ADC_ENORC

Enable Out Of Range Comparator Register(D3_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
ENORC7	ENORC6	ENORC5	ENORC4	ENORC3	ENORC2	ENORC1	ENORC0
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENORCx (x = 0 - 7)	x	rw	Enable Out of Range Comparator x This bit defines if the out of range comparator is enabled in the corresponding analog input channel. 0 _B Out of range comparator disabled. 1 _B Out of range comparator enabled. <i>Note: Bit 4-7 are only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>

Analog to Digital Converter

The bit fields in this register selects for detection of voltages higher or lower than Vddp at each input ADC channels that triggers the out of range comparator.

ADC_CORC

Configure Out Of Range Comparator Register(D2_H)

Reset Value: 00_H

RMAP: 0, PAGE: 4

7	6	5	4	3	2	1	0
CNF7	CNF6	CNF5	CNF4	CNF3	CNF2	CNF1	CNF0
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CNF _x (x = 0 - 7)	x	rw	Out of Range Comparator Flag x 0 _B Detect voltage lower than Vddp at analog input pin. 1 _B Detect voltage higher than Vddp at analog input pin. <i>Note: Bit 4-7 are only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>

Analog to Digital Converter

The bit fields in these registers indicates if a voltage out of range have occurred for the corresponding analog input channels. If a voltage out of range have occurred it will be latched to 1 in this register. The event flag can be cleared by writing a 0 to this register.

ADC_LORC

Latched Out Of Range Event Register (D2_H)

Reset Value: 00_H

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
LORE7	LORE6	LORE5	LORE4	LORE3	LORE2	LORE1	LORE0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
LORE _x (x = 0 - 7)	x	rwh	Latched Out of Range Event x This bit defines which analog input channel voltage out of range event have occurred. The LORE bit is set by hardware and can only be cleared by software. 0 _B No voltage out of range event for channel x has not occurred. Writing a "0" clears this register. 1 _B A voltage out of range event for channel x has occurred. <i>Note: Bit 4-7 are only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>

21.8.6 Conversion Timing

The total time required for a conversion depends on several user-definable factors:

- The ADC conversion clock frequency, where $f_{\text{ADCI}} = f_{\text{ADC}} / (\text{CTC} + 3)$
- The selected sample time, where $t_s = (2 + \text{STC}) \times t_{\text{ADCI}}$
(STC = additional sample time, see also [Section 21.5.1](#))
- The selected result width N (8/10 bits)
- Synchronization steps done at module clock speed

The conversion time is the sum of sample time, conversion steps, and synchronization. It can be computed with the following formula:

$$t_{\text{CN}} = t_{\text{ADC}} \times (1 + r \times (3 + n + \text{STC}))$$

$$r = \text{CTC} + 3,$$

CTC = Conversion Time Control ([ADC_GLOBCTR.CTC](#))

STC = Sample Time Control ([ADC_INPCR0.STC](#))

n = 8 or 10 (for 8-bit and 10bit conversion respectively)

$$t_{\text{ADC}} = 1 / f_{\text{ADC}}$$

The frequency at which conversions are triggered also depends on several configurable factors:

- The selected conversion time, according to the input class definitions.
- Delays induced by cancelled conversions that must be repeated.
- The configured arbitration cycle time.
- The frequency of external trigger signals, if enabled.

Example1:

When N=10bit, STC=0000_B, $f_{\text{ADC}}=48\text{Mhz}$, CTC=01_B

$$f_{\text{ADCI}} = (48\text{Mhz} / (1 + 3)) = 12\text{Mhz} = 83.33\text{ns} \quad (21.1)$$

$$t_{\text{sample}} = (2 + 0) \times 83.33\text{ns} = 166.67\text{ns} \quad (21.2)$$

Table 21-6 Sample Time

Sample Time	ADC_INPCR0.STC (bin)
$2 \times t_{\text{ADCl}}$	0000 _B
$3 \times t_{\text{ADCl}}$	0001 _B
...	
$17 \times t_{\text{ADCl}}$	1111 _B

$$t_{\text{conv}} = \left(\frac{1}{48\text{Mhz}} \right) \times (1 + (1 + 3) \times (3 + 10 + 0)) = 1.104\mu\text{s} \quad (21.3)$$

21.8.7 Channel Events and Interrupts

A channel event interrupt can be generated based on a channel event according to the structure shown in [Figure 21-20](#). If a channel event is detected, it sets the corresponding indication flag in register [ADC_CHINFR](#). The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register [ADC_CHINCR](#).

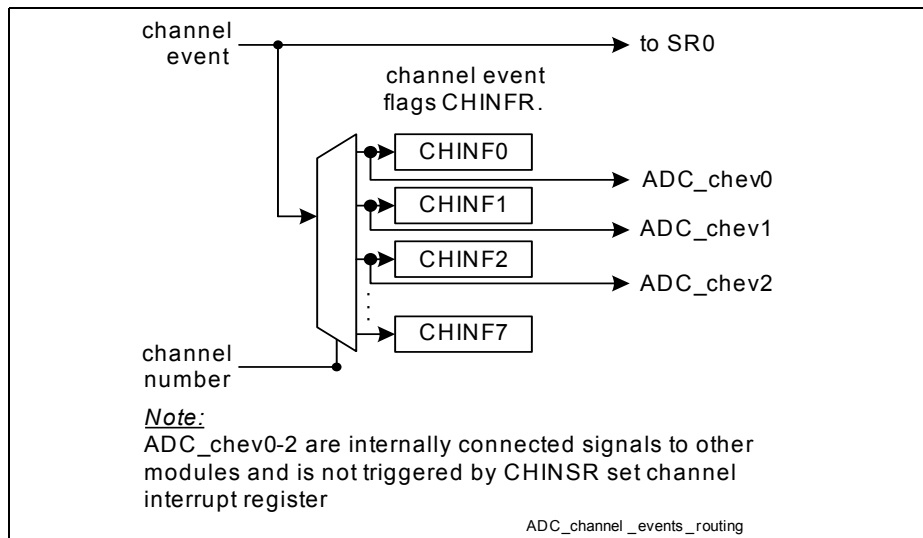


Figure 21-20 Channel Event Interrupt Generation

21.9 Conversion Result Handling

The conversion results of each analog input channel can be stored in one of 4 conversion result registers (selected by bitfield RESRSEL in the associated channel control register CHCTRx). This structure provides different locations for the conversion results of different groups of channels. Depending on the application needs (data reduction, auto-scan, alias feature, etc.), the user can distribute the conversion results to minimize CPU load or to be more tolerant against interrupt latency.

21.9.1 Storage of Conversion Results

Each result register has an individual data valid flag (VFX) associated with it. This flag indicates when “new” valid data has been stored in the corresponding result register and can be read out.

Depending of the result register read view (see below), the corresponding valid flag is automatically cleared when the result is read or remains set.

- Automatically clearing the valid flag provides an easy handshake between result generation and retrieval. This also supports wait-for-read mode.
- Leaving the valid flag set supports debugging by delivering the result value without disturbing the handshake with the application.

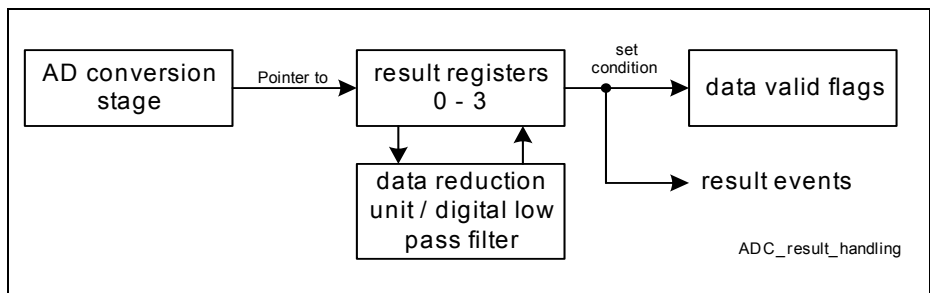


Figure 21-21 Conversion Result Storage

Conversion result handling comprises the following functions:

- Storage of conversion results to user-configurable registers
- Wait-for-read mode to avoid loss of data if several channels share one result register (see [Section 21.10](#))
- Result event interrupts (see [Section 21.10.1](#))
- Data reduction or anti-aliasing filtering and Digital low pass filtering (see [Section 21.10.2](#))

Analog to Digital Converter

Up to 4 result values can be added in each result register. This reduces the frequency of interrupts generated by the ADC.

- Standard application read view (RESRxL/H):
 - **8bit conversion mode with Data Reduction and Digital Low Pass Filter disabled.** (i.e. **ADC_GLOBCTR.DW=1**, **RCRx.DRCTR=0**, **RCRx.DLPF=0**)
 When this mode is chosen, the targetted result register in **ADC_RESRxL (x=0-3)** and **ADC_RESRxH (x=0-3)** would be shown as **Figure 21-22**.
 In **ADC_RESRxL (x=0-3)**, bits 2-0 indicate the channel number whose conversion triggered the result event and in **ADC_RESRxH (x=0-3)** bits 7-0 returns the conversion result. Reading the result automatically clears the valid flag. This view is useful only without data reduction.
 - **10bit conversion mode with Data Reduction and Digital Low Pass Filter disabled.** (i.e. **ADC_GLOBCTR.DW=0**, **RCRx.DRCTR=0**, **RCRx.DLPF=0**)
 When this mode is chosen, the targetted result register in **ADC_RESRxL (x=0-3)** and **ADC_RESRxH (x=0-3)** would be shown as **Figure 21-23**.
 In **ADC_RESRxL (x=0-3)**, bits 2-0 indicate the channel number whose conversion triggered the result event, bits 7-5 return the last 3bits of the conversion result. In **ADC_RESRxH (x=0-3)** bits 6-0 returns bit 9-3 of the conversion result. Reading the result automatically clears the valid flag. This view is useful only without data reduction
- Accumulated application read view (RESRxL/H):
 - **8bit conversion mode with Data Reduction Enabled and Digital Low Pass Filter disabled.** (i.e. **ADC_GLOBCTR.DW=1**, **RCRx.DRCTR=1**, **RCRx.DLPF=0**)
 When this mode is chosen, the targetted result register in **ADC_RESRxL (x=0-3)** and **ADC_RESRxH (x=0-3)** would be shown as **Figure 21-24**.
 After the 1st conversion, in **ADC_RESRxL (x=0-3)** result register, bits 2-0 indicate the channel number whose conversion triggered the result event. In **ADC_RESRxH (x=0-3)** bits 7-0 returns the conversion result. This allows the result to be read in a single read. Reading the result automatically clears the valid flag.
 At the 2nd conversion onwards, in **ADC_RESRxL (x=0-3)** result register, bits 2-0 indicate the channel number whose conversion triggered the result event , bit 7 indicate the last bit of the accumulated result. In **ADC_RESRxH (x=0-3)** bits 7-0 returns bit 8-1 of the accumulated result.
 - **10bit conversion mode with Data Reduction Enabled and Digital Low Pass Filter disabled.** (i.e. **ADC_GLOBCTR.DW=0**, **RCRx.DRCTR=1**, **RCRx.DLPF=0**)
 When this mode is chosen, the targetted result register in **ADC_RESRxL (x=0-3)** and **ADC_RESRxH (x=0-3)** would be shown as **Figure 21-25**.
 After the 1st conversion, in **ADC_RESRxL (x=0-3)** result register, bits 2-0 indicate the channel number whose conversion triggered the result event, bits 7-5 returns bits 2-0 of the conversion results. In **ADC_RESRxH (x=0-3)** bits 6-0 returns bits 9-3 of the conversion result. Reading the result automatically clears the valid flag.
 At the 2nd conversion onwards, in **ADC_RESRxL (x=0-3)** result register, bits 2-0

Analog to Digital Converter

indicate the channel number whose conversion triggered the result event , bit 7-5 returns bits 2-0 of the accumulated result. In ADC_RESRxH (x=0-3) bits 7-0 returns bit 10-3 of the accumulated result.

- Digital low pass filtered application read view (RESRxL/H):
10bit conversion mode with Data Reduction Disabled and Digital Low Pass Filter enabled. (i.e **ADC_GLOBCTR.DW=0**, **RCRx.DRCTR=0**, **RCRx.DLPF=1**) This mode is only available for 10bit conversions and the Data Reduction Feature cannot be turned on when the Digital Low Pass Filter is being used, since there is only one adder.

When this mode is chosen, the targetted result register in ADC_RESRxL (x=0-3) and ADC_RESRxH (x=0-3) would be shown as **Figure 21-26**.

In ADC_RESRxL (x=0-3), bits 2-0 indicate the channel number whose conversion triggered the result event, bits 7-3 returns bits 4-0 of the filtered result. In ADC_RESRxH (x=0-3) bits 7-0 returns bits 12-5 of the filtered result.

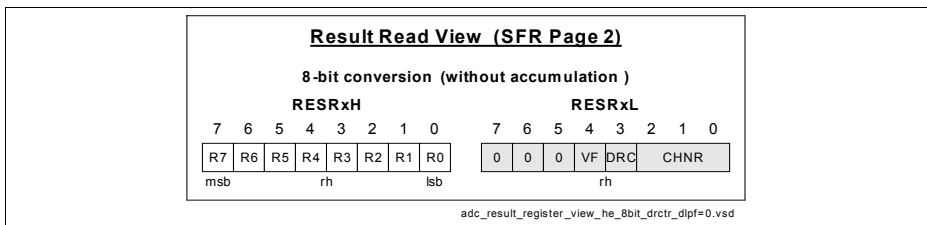


Figure 21-22 8bit, RCRx.DRCTR=0, RCRx.DLPF=0, Result Register View

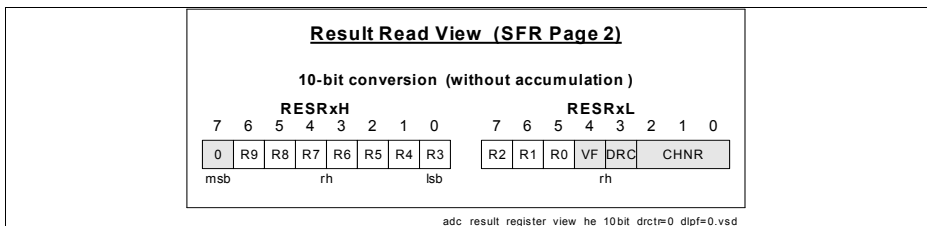


Figure 21-23 10bit, RCRx.DRCTR=0, RCRx.DLPF=0, Result Register View

Analog to Digital Converter

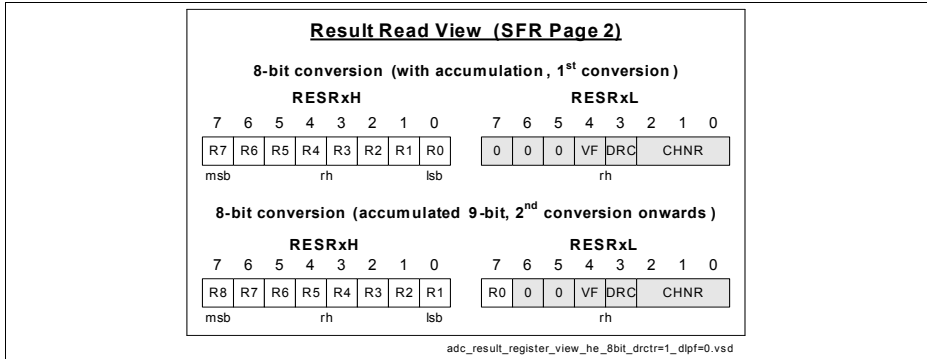


Figure 21-24 8bit, RCRx.DRCTR=1, RCRx.DLPF=0, Result Register View

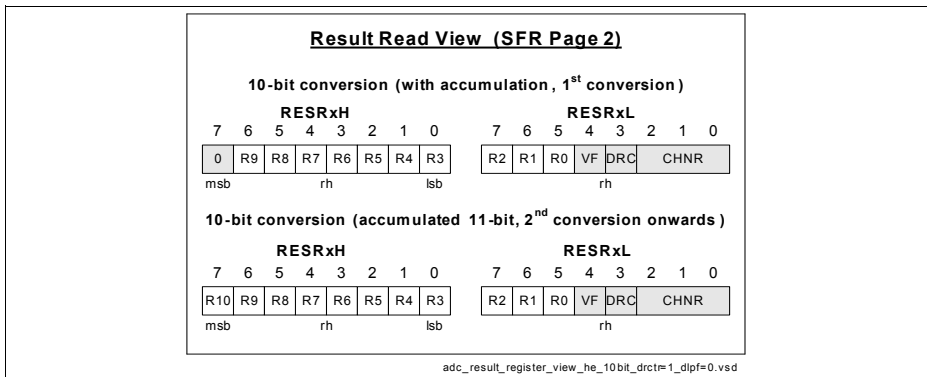


Figure 21-25 10bit, RCRx.DRCTR=1, RCRx.DLPF=0, Result Register View

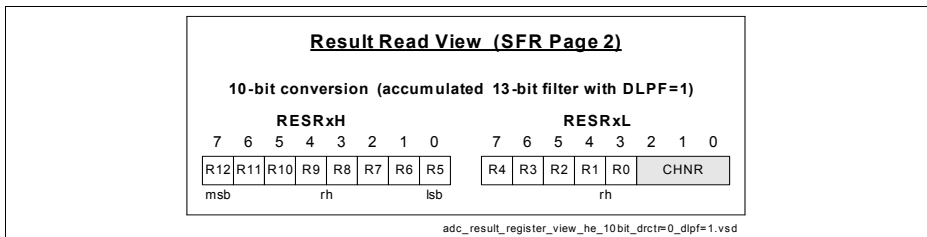


Figure 21-26 10bit, RCRx.DRCTR=0, RCRx.DLPF=1, Result Register View

All conversion results are stored in the result registers, the content of the result register is available at a single page address and data are aligned according to the different modes being selected.

Analog to Digital Converter

The standard read views of the result registers consists of ADC_RESR_{xL} and ADC_RESR_{xH} which delivers the conversion result and the related channel number. The corresponding valid flag is cleared when register RESR_x is read (application view).

ADC_RESR_{xL} (x = 0 - 2)

Result Register x Low, View Std. 8-bit or Acc. 8-bit 1st conv.(CA_H + x * 2) Reset Value: 00_H

ADC_RESR3L

Result Register 3 Low, View Std. 8-bit or Acc. 8-bit 1st conv.(D2_H) Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0			VF	DRC	CHNR		
r			rh	rh	rh		

Field	Bits	Type	Description
CHNR	[2:0]	rh	Channel Number This bit field contains the channel number of the latest register update. <i>Note: Bit 2 is only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>
DRC	3	rh	Data Reduction Counter This bit field indicates how many conversion results have still to be accumulated to generate the final result for data reduction. 0 _B The final result is available in the result register. The valid flag is automatically set when this bit field is set to 0. 1 _B 1 more conversion result must be added to obtain the final result in the result register. The valid flag is automatically reset when this bit field is set to 1.

Analog to Digital Converter

Field	Bits	Type	Description
VF	4	rh	Valid Flag for Result Register x This bit indicates that the contents of the result register x are valid. 0_B The result register x does not contain valid data. 1_B The result register x contains valid data.
0	[7:5]	r	Reserved Returns 0 if read; should be written with 0.

ADC_RESRxH (x = 0 - 2)

Result Register x High, View Std. 8-bit or Acc. 8-bit 1st conv.(CB_H + x * 2) Reset Value: 00_H

ADC_RESR3H

Result Register 3 High, View Std. 8-bit or Acc. 8-bit 1st conv.(D3_H) Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
RESULT[7:0]							
rh							

Field	Bits	Type	Description
RESULT[7:0]	[7:0]	rh	Conversion Result This bit field contains the conversion result or the result of the data reduction filter.

ADC_RESRxL (x = 0 - 2)

Result Register x Low, View Std. 10-bit(CA_H + x * 2) Reset Value: 00_H

ADC_RESR3L

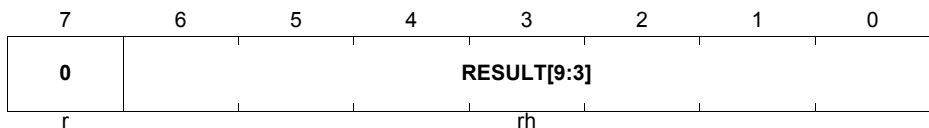
Result Register 3 Low, View Std.10-bit(D2_H) Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
RESULT[2:0]			VF	DRC	CHNR		
rh			rh	rh	rh		

Analog to Digital Converter

Field	Bits	Type	Description
CHNR	[2:0]	rh	Channel Number This bit field contains the channel number of the latest register update. <i>Note: Bit 2 is only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>
DRC	3	rh	Data Reduction Counter This bit field indicates how many conversion results have still to be accumulated to generate the final result for data reduction. 0_B The final result is available in the result register. The valid flag is automatically set when this bit field is set to 0. 1_B 1 more conversion result must be added to obtain the final result in the result register. The valid flag is automatically reset when this bit field is set to 1.
VF	4	rh	Valid Flag for Result Register x This bit indicates that the contents of the result register x are valid. 0_B The result register x does not contain valid data. 1_B The result register x contains valid data.
RESULT[2:0]	[7:5]	rh	Conversion Result This bit field contains the conversion result or the result of the data reduction filter.

ADC_RESRxH (x = 0 - 2)
Result Register x High, View Std. 10-bit($CB_H + x * 2$)
Reset Value: 00_H
ADC_RESR3H
Result Register 3 High, View Std. 10-bit(D3_H)
Reset Value: 00_H
RMAP: 0, PAGE: 2


Analog to Digital Converter

Field	Bits	Type	Description
RESULT[9:3]	[6:0]	rh	Conversion Result This bit field contains the conversion result or the result of the data reduction filter.
0	7	r	Reserved Returns 0 if read; should be written with 0.

ADC_RESRxL (x = 0 - 2)

Result Register x Low, View Acc. 8-bit 2nd conv.(CA_H + x * 2) Reset Value: 00_H

ADC_RESR3L

Result Register 3 Low, View Acc. 8-bit 2nd conv.(D2_H) Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
R0	0	VF	DRC	CHNR			
rh	r	rh	rh	rh			

Field	Bits	Type	Description
CHNR	[2:0]	rh	Channel Number This bit field contains the channel number of the latest register update. <i>Note: Bit 2 is only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>
DRC	3	rh	Data Reduction Counter This bit field indicates how many conversion results have still to be accumulated to generate the final result for data reduction. 0_B The final result is available in the result register. The valid flag is automatically set when this bit field is set to 0. 1_B 1 more conversion result must be added to obtain the final result in the result register. The valid flag is automatically reset when this bit field is set to 1.

Analog to Digital Converter

Field	Bits	Type	Description
VF	4	rh	Valid Flag for Result Register x This bit indicates that the contents of the result register x are valid. 0_B The result register x does not contain valid data. 1_B The result register x contains valid data.
0	[6:5]	r	Reserved Returns 0 if read; should be written with 0.
RESULT[0]	7	rh	Conversion Result This bit field contains the conversion result or the result of the data reduction filter.

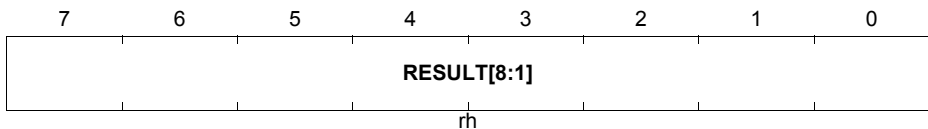
ADC_RESRxH (x = 0 - 2)

Result Register x High, View Acc. 8-bit 2nd conv.(CB_H + x * 2) Reset Value: 00_H

ADC_RESR3H

Result Register 3 High, View Acc. 8-bit 2nd conv.(D3_H) Reset Value: 00_H

RMAP: 0, PAGE: 2



Field	Bits	Type	Description
RESULT[8:1]	[7:0]	rh	Conversion Result This bit field contains the conversion result or the result of the data reduction filter.

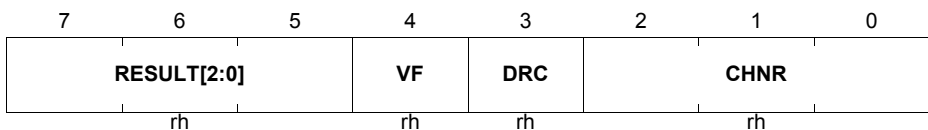
ADC_RESRxL (x = 0 - 2)

Result Register x Low, View Acc. 10-bit 1st conv.(CA_H + x * 2) Reset Value: 00_H

ADC_RESR3L

Result Register 3 Low, View Acc. 10-bit 1st conv.(D2_H) Reset Value: 00_H

RMAP: 0, PAGE: 2



Analog to Digital Converter

Field	Bits	Type	Description
CHNR	[2:0]	rh	Channel Number This bit field contains the channel number of the latest register update. <i>Note: Bit 2 is only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>
DRC	3	rh	Data Reduction Counter This bit field indicates how many conversion results have still to be accumulated to generate the final result for data reduction. 0_B The final result is available in the result register. The valid flag is automatically set when this bit field is set to 0. 1_B 1 more conversion result must be added to obtain the final result in the result register. The valid flag is automatically reset when this bit field is set to 1.
VF	4	rh	Valid Flag for Result Register x This bit indicates that the contents of the result register x are valid. 0_B The result register x does not contain valid data. 1_B The result register x contains valid data.
RESULT[2:0]	[7:5]	rh	Conversion Result This bit field contains the conversion result or the result of the data reduction filter.

ADC_RESRxH (x = 0 - 2)
Result Register x High, View Acc. 10-bit 1st conv. ($CB_H + x * 2$) Reset Value: 00_H
ADC_RESR3H
Result Register 3 High, View Acc. 10-bit 1st conv. ($D3_H$) Reset Value: 00_H
RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0	RESULT[9:3]						
r	rh						

Analog to Digital Converter

Field	Bits	Type	Description
RESULT[9:3]	[6:0]	rh	Conversion Result This bit field contains the conversion result or the result of the data reduction filter.
0	7	r	Reserved Returns 0 if read; should be written with 0.

ADC_RESRxL (x = 0 - 2)

Result Register x Low, View Acc. 10-bit 2nd conv.(CA_H + x * 2) Reset Value: 00_H

ADC_RESR3L

Result Register 3 Low, View Acc. 10-bit 2nd conv.(D2_H) Reset Value: 00_H

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
RESULT[2:0]			VF	DRC	CHNR		
rh			rh	rh	rh		

Field	Bits	Type	Description
CHNR	[2:0]	rh	Channel Number This bit field contains the channel number of the latest register update. <i>Note: Bit 2 is only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>
DRC	3	rh	Data Reduction Counter This bit field indicates how many conversion results have still to be accumulated to generate the final result for data reduction. 0 _B The final result is available in the result register. The valid flag is automatically set when this bit field is set to 0. 1 _B 1 more conversion result must be added to obtain the final result in the result register. The valid flag is automatically reset when this bit field is set to 1.

Analog to Digital Converter

Field	Bits	Type	Description
VF	4	rh	Valid Flag for Result Register x This bit indicates that the contents of the result register x are valid. 0_B The result register x does not contain valid data. 1_B The result register x contains valid data.
RESULT[2:0]	[7:5]	rh	Conversion Result This bit field contains the conversion result or the result of the data reduction filter.

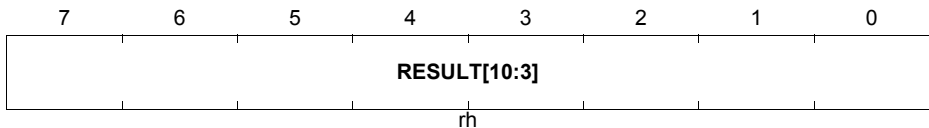
ADC_RESRxH (x = 0 - 2)

Result Register x High, View Acc. 10bit 2nd conv.($CB_H + x * 2$) Reset Value: 00_H

ADC_RESR3H

Result Register 3 High, View Acc. 10bit 2nd conv.($D3_H$) Reset Value: 00_H

RMAP: 0, PAGE: 2



Field	Bits	Type	Description
RESULT[10:3]	[7:0]	rh	Conversion Result This bit field contains the conversion result or the result of the data reduction filter.

ADC_RESRxL (x = 0 - 2)

Result Register x Low, View LPF. 10-bit($CA_H + x * 2$) Reset Value: 00_H

ADC_RESR3L

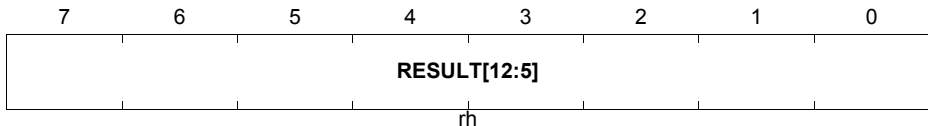
Result Register 3 Low, View LPF. 10-bit($D2_H$) Reset Value: 00_H

RMAP: 0, PAGE: 2



Analog to Digital Converter

Field	Bits	Type	Description
CHNR	[2:0]	rh	Channel Number This bit field contains the channel number of the latest register update. <i>Note: Bit 2 is only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>
RESULT[4:0]	[7:3]	rh	Conversion Result This bit field contains the result of the digital low pass filter.

ADC_RESRxH (x = 0 - 2)
Result Register x High, View LPF, 10-bit($CB_H + x * 2$)
Reset Value: 00_H
ADC_RESR3H
Result Register 3 High, View LPF, 10-bit(D3_H)
Reset Value: 00_H
RMAP: 0, PAGE: 2


Field	Bits	Type	Description
RESULT[12:5]	[7:0]	rh	Conversion Result This bit field contains the result of the digital low pass filter.

Analog to Digital Converter

The Result Control Registers control the behavior of the result registers and monitor their status.

ADC_RCRx (x = 0 - 3)

Result Control Register x

(CA_H + x * 1)

Reset Value: 00_H

RMAP: 0, PAGE: 4

7	6	5	4	3	2	1	0
VFCTR	WFR	0	IEN	0	DLPF	0	DRCTR
rw	rw	r	rw	r	rw	r	rw

Field	Bits	Type	Description
DRCTR ¹⁾	0	rw	Data Reduction Control This bit defines how many conversion results are accumulated for data reduction. It defines the reload value for bit DRC. 0 _B The data reduction filter is disabled. The reload value for DRC is 0, so the accumulation is done over 1 conversion. 1 _B The data reduction filter is enabled. The reload value for DRC is 1, so the accumulation is done over 2 conversions.
DLPF ¹⁾	2	rw	Digital Low Pass Filter Control This bit defines if the conversion results is processed through the Digital low pass filter. 0 _B The digital low pass filter is disabled. 1 _B The digital low pass filter is enabled and previous result register is cleared for the first time when the bit is changed from 0->1. See Page 21-82
IEN	4	rw	Interrupt Enable This bit enables the event interrupt related to the result register x. An event interrupt can be generated when DRC is set to 0 (after decrementing or by reload). 0 _B The event interrupt is disabled. 1 _B The event interrupt is enabled.

Analog to Digital Converter

Field	Bits	Type	Description
WFR	6	rw	Wait-for-Read Mode This bit enables the wait-for-read mode for result register x. 0 _B The wait-for-read mode is disabled. 1 _B The wait-for-read mode is enabled.
VFCTR	7	rw	Valid Flag Control This bit enables the reset of valid flag (by read access to high byte) for result register x. 0 _B VF unchanged by read access to RESR _x H/RESR _A xH. (default) 1 _B VF reset by read access to RESR _x H/RESR _A xH.
0	1,3,5	r	Reserved Returns 0 if read; should be written with 0.

1) For 10bit conversion, either Data Reduction Filter OR Digital Low Pass Filter can be used. However since there is only one adder both filter cannot be turned on at the same time. Thus when both filter is turned on at the same time, DRCTR=1 and DLPF=1, the results would be the same as DRCTR=0 and DLPF=1, digital low pass filter enabled.

For 8-bit conversion, only Data Reduction Filter can be used. When 8bit resolution mode is chosen and digital low pass filter is enabled DLPF=1 the results will be the same as DLPF=0,digital low pass filter disabled.

Analog to Digital Converter

The Valid Flag Register summarizes the flags indicating that the corresponding result register contents are valid.

ADC_VFCR

Valid Flag Clear Register

(CE_H)

Reset Value: 00_H

RMAP: 0, PAGE: 4

7	6	5	4	3	2	1	0
0				VFC3	VFC2	VFC1	VFC0
r				w	w	w	w

Field	Bits	Type	Description
VFCx(x = 0 - 3)	x	w	Clear Valid Flag for Result Register x 0 _B No action 1 _B Bit VF in result register RESR _x L is reset.
0	[7:4]	r	Reserved Returns 0 if read; should be written with 0.

21.10 Wait-for-Read Mode

The wait-for-read mode is a feature to prevent data loss due to overwriting a result register with a new conversion result before the CPU has read the previous data. For example, auto-scan conversion sequences or other sequences with “relaxed” timing requirements are likely to use the same result register. However, the results come from different input channels, so an overwrite would destroy the result from the previous conversion¹⁾.

Wait-for-read mode automatically suspends the start of a conversion for this channel until the current result has been read. So a conversion or a conversion sequence can be requested by a hardware or software trigger, while each conversion is only started after the previous one has been read. This automatically aligns the conversion sequence with the CPU capability to read the formerly converted result (interrupt latency).

If wait-for-read mode is enabled for a result register (bit WFR = 1 in the corresponding result control register), a request source does not generate a conversion request while the targeted result register contains valid data (indicated by the valid flag VF_x = 1) or if a currently running conversion targets the same result register.

1) Repeated conversions of a single channel that use a separate result register will not destroy other results, but rather update their own previous result value. This way, always the actual signal data is available in the result register.

Analog to Digital Converter

If two request sources target the same result register with wait-for-read mode selected, a higher priority source cannot interrupt a lower priority conversion request started before the higher priority source has requested its conversion. Cancel-inject-repeat mode does not work in this case. If the higher priority request targets a different result register, the lower priority conversion can be cancelled and repeated afterwards.

21.10.1 Result Events and Interrupts

A result event interrupt can be generated based on a result event according to the structure shown in [Figure 21-27](#). If a result event is detected, it sets the corresponding indication flag in register [ADC_EVINFR](#). These flags can also be set by writing a 1 to the corresponding bit position at [ADC_EVINSR](#), whereas writing 0 has no effect. The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register [ADC_EVINCR](#).

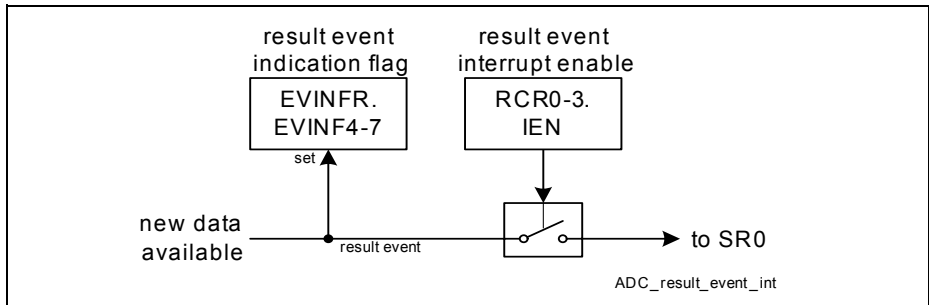


Figure 21-27 Result Event Interrupt Generation

The Request Source Event Interrupt, Channel Event Interrupt and Result Event Interrupt all share the same service request output line SR0.

The result events and the request source events share the same registers. The result events are located at the following bit positions in register [ADC_EVINFR](#):

- Event 4: Result event of result register 0.
- Event 5: Result event of result register 1.
- ...
- Event 7: Result event of result register 3.

21.10.2 Data Reduction and Filtering

Data reduction automatically accumulates a series of conversion results before generating a result interrupt. This can remove some noise from the input signal and reduces the CPU load required to unload the conversion data from the ADC.

The standard data reduction mode accumulates result values within arbitrary result registers.

Analog to Digital Converter

- If DRC becomes 0, either decremented from 1 (t_2 , t_4 , t_6 , t_8 in the example) or loaded from DRCTR, the valid bit for the respective result register is set and a result register event occurs.

The final result must be read before the next data reduction sequence starts (before t_3 , t_5 , t_7 or t_9 in the example). This automatically clears the valid flag.

Digital Low Pass Filter Mode

Alternatively, the data reduction logic can build a digital low-pass filter by adding the amplified result (factor 4) to the attenuated current value (factor 0.5)

Each result register can be individually enabled for low-pass filtering, controlled by bitfield LPFEN in registers **ADC_RCRx** ($x = 0 - 3$). The data reduction counter DRC is not used in this case.

Note: For low-pass filtering, the result data must come from one dedicated channel.

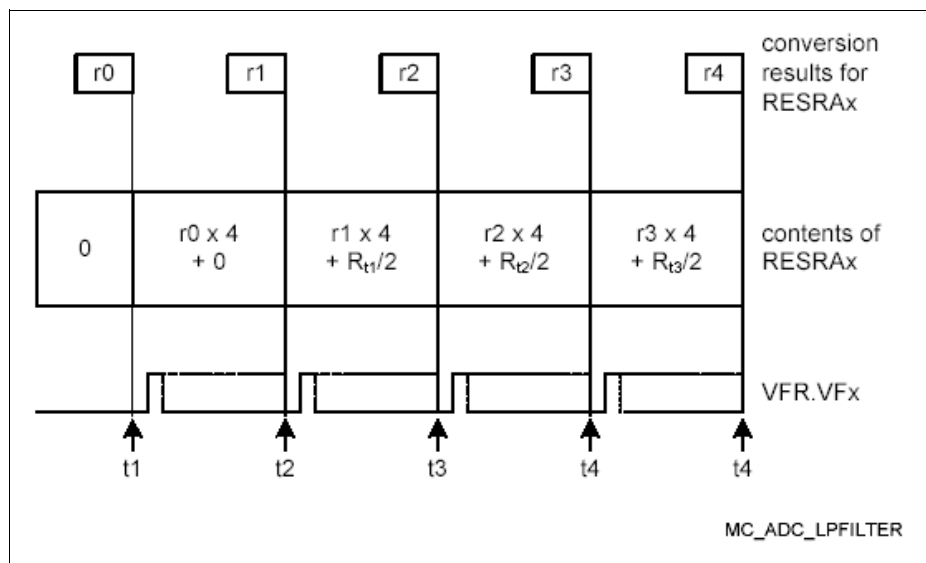


Figure 21-29 Low-Pass Filter Operation

Each result is multiplied by 4 and added to half the previous register value. The valid flag is activated after the addition.

21.11 Interrupt Request Handling

Interrupts can be generated by several types of events. Each ADC kernel provides 2 service request output signals (ADCx_SR[1:0]) connected to interrupt nodes. Four types of events can generate interrupt requests:

- **Request source events:** indicate that a request source completed the requested conversion sequence. For a scan source, the event is generated when the complete defined set of channels has been arbitrated. For a sequential source, the user can define where inside a conversion sequence a request source event is generated. Request source events indicate that a conversion sequence has reached a defined state and software can access the related set of results.
- **Channel events:** indicate that a conversion is finished. Optionally, channel events can be generated only for conversion result within a programmable value range. Channel events preferably indicate analog input values inside or outside a nominal operating range. This offloads the CPU load from background tasks, i.e. an interrupt is only required if the specified conversion result range is met or exceeded.
- **Result events:** indicate a new valid result in a result register. Usually, this triggers a read action by the CPU. Optionally, result events can be generated only at a reduced rate if data reduction or digital low pass filter is active.
- **Out of range comparator events:** indicate that a voltage higher or lower than V_{ddp} is detected at the analog input channels.

Each ADC event is indicated by a dedicated flag that can be cleared by software. If an interrupt is enabled for a certain event, the interrupt is generated for each event, independent of the status of the corresponding event indication flag. This ensures efficient handling of ADC events (the ADC event can generate an interrupt without the need to clear the indication flag).

The Request source event, Channel event and Result events share a common service request output, ADC_SR0, while the Out of range comparator events uses the ADC_SR1 service request output.

Note: A conversion can lead to three interrupts, one of each type, if all are enabled.

In this case, the ADC module first triggers the request source event interrupt, then the channel event interrupt, followed by the result event interrupt (all within a few f_{ADC} clock cycles).

Analog to Digital Converter

The Event indication Flag Register ADC_EVINFR monitors both the detected request source events (flags EVINF0-EVINF1) and the result events (flags EVINF4-EVINF7).

ADC_EVINFR

Event Interrupt Flag Register

(CE_H)

Reset Value: 00_H

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
EVINF7	EVINF6	EVINF5	EVINF4	0		EVINF1	EVINF0
rh	rh	rh	rh	r		rh	rh

Field	Bits	Type	Description
EVINF0, EVINF1, EVINF4, EVINF5, EVINF6, EVINF7	0, 1, 4, 5, 6, 7	rh	Interrupt Flag for Event x This bit monitors the status of the event interrupt x. 0 _B An event interrupt for event x has not occurred. Writing a “0” clears this register. 1 _B An event interrupt for event x has occurred. Writing a “1” sets this register and generates an interrupt pulse (if interrupt is enabled).
0	[3:2]	r	Reserved Returns 0 if read; should be written with 0.

Analog to Digital Converter

The Event Interrupt Set Flag Register **ADC_EVINSR** sets the corresponding bit at **ADC_EVINFR** and generates an interrupt request when the associated bit is written.

ADC_EVINSR

Event Interrupt Set Flag Register (D2_H)

Reset Value: 00_H

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
EVINS7	EVINS6	EVINS5	EVINS4	0		EVINS1	EVINS0
w	w	w	w	r		w	w

Field	Bits	Type	Description
EVINS0, EVINS1, EVINS4, EVINS5, EVINS6, EVINS7	0, 1, 4, 5, 6, 7	w	Set Interrupt Flag for Event x 0 _B No action 1 _B Bit EVINFR.x is set.
0	[3:2]	r	Reserved Returns 0 if read; should be written with 0.

*Note: Writing 1 to a bit of the Event Interrupt Set Flag Register **ADC_EVINSR** sets the corresponding bit at **ADC_EVINFR** and generates the associated interrupt request. Writing a 0 has not effect.*

Analog to Digital Converter

Writing a 1 to a bit position in the Event Indication Clear Register **ADC_EVINCR** clears the corresponding event indication flag EVINFRx in register **ADC_EVINFR**. If a request source or result event is detected when the corresponding bit position is written with a 1, flag EVINFRx is cleared.

ADC_EVINCR

Event Interrupt Clear Flag Register (CF_H)

Reset Value: 00_H

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
EVINC7	EVINC6	EVINC5	EVINC4	0		EVINC1	EVINC0
W	W	W	W	r		W	W

Field	Bits	Type	Description
EVINC0, EVINC1, EVINC4, EVINC5, EVINC6, EVINC7	0, 1, 4, 5, 6, 7	w	Clear Interrupt Flag for Event x 0 _B No action 1 _B Bit EVINFR.x is reset.
0	[3:2]	r	Reserved Returns 0 if read; should be written with 0.

Analog to Digital Converter

The Channel Event Indication Flag Register CHINFR monitors the detected channel events for channels 0 ... 7

ADC_CHINFR

Channel Interrupt Flag Register

(CA_H)

Reset Value: 00_H

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
CHINF7	CHINF6	CHINF5	CHINF4	CHINF3	CHINF2	CHINF1	CHINF0
rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
CHINFx (x = 0 - 7)	x	rh	Interrupt Flag for Channel x This bit monitors the status of the channel interrupt x. 0 _B A channel interrupt for channel x has not occurred. 1 _B A channel interrupt for channel x has occurred. <i>Note: Bit 4-7 are only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>

Analog to Digital Converter

Writing 1 to a bit of the CHINSR register sets the corresponding bit and generates the associated interrupt request. Writing a 0 has no effect

ADC_CHINSR

Channel Interrupt Set Register

(CC_H)

Reset Value: 00_H

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
CHINS7	CHINS6	CHINS5	CHINS4	CHINS3	CHINS2	CHINS1	CHINS0
W	W	W	W	W	W	W	W

Field	Bits	Type	Description
CHINSx (x = 0 - 7)	x	w	Set Interrupt Flag for Channel x 0 _B No action 1 _B Bit CHINFR.x is set and an interrupt pulse is generated. <i>Note: Bit 4-7 are only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>

Analog to Digital Converter

Writing a 1 to a bit position in the channel indication clear register CHINCR clears the corresponding channel event indication flag CHINFRx in register **ADC_CHINFR**. If a channel event is detected when the corresponding bit position is written with a 1, flag CHINFRx is cleared.

ADC_CHINCR

Channel Interrupt Clear Register (CB_H)

Reset Value: 00_H

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
CHINC7	CHINC6	CHINC5	CHINC4	CHINC3	CHINC2	CHINC1	CHINC0
W	W	W	W	W	W	W	W

Field	Bits	Type	Description
CHINCx (x = 0 - 7)	x	w	Clear Interrupt Flag for Channel x 0 _B No action 1 _B Bit CHINFR.x is reset. <i>Note: Bit 4-7 are only applicable for devices that have 8 ADC channels. For channels not implemented, these bits should be treated as Reserved bits of type 'r' which returns '0' if read and should be written with 0.</i>

21.12 Register Mapping

The addresses of the kernel SFRs are listed in **Table 21-7** and **Table 21-8**.

Table 21-7 SFR Address List for Pages 0 – 3

Address	Page 0	Page 1	Page 2	Page 3
CAH	ADC_GLOBCTR	CHCTR0	RESR0L	
CBH	ADC_GLOBSTR	CHCTR1	RESR0H	
CCH	ADC_PRAR	CHCTR2	RESR1L	
CDH	ADC_LCBR0	CHCTR3	RESR1H	
CEH	ADC_INPCR0	CHCTR4	RESR2L	
CFH	ADC_LCBR1	CHCTR5	RESR2H	
D2H	ADC_LORC	CHCTR6	RESR3L	
D3H	ADC_ENORC	CHCTR7	RESR3H	

Table 21-8 SFR Address Listing for Pages 4 – 7

Address	Page 4	Page 5	Page 6	Page 7
CAH	RCR0	ADC_CHINFR	ADC_CRCCR1	
CBH	RCR1	ADC_CHINCR	ADC_CRPR1	
CCH	RCR2	ADC_CHINSR	ADC_CRM1	
CDH	RCR3		ADC_QMR0	
CEH	ADC_VFCR	ADC_EVINFR	ADC_QSR0	
CFH	ADC_ALR0	ADC_EVINCR	ADC_Q0R0	
D2H	ADC_CORC	ADC_EVINSR	ADC_QBUR0/ ADC_QINR0	
D3H	ADC_ETRCR			

22 Boot ROM User Routines

The Boot ROM User Routines provides a set of useful routines that can be called by user application. These routines have been developed as part of the Boot ROM functionality and are provided to the user via a Boot ROM User Routine table. User application would be able to call these routines provided in [Table 22-1](#).

Table 22-1 Boot ROM User Routine table

Address	Name	Description
0xDFDE	BR_FLASH_READ_MODE_STATUS	Check the Read Mode status of the selected Flash Bank
0xDFE1	BR_GET_4_BYTES_INFO	Read the 4 bytes of Chip Identification Number and User Identification Number (USER_ID)
0xDFE4	BR_PROG_USER_ID	Program 4 bytes of USER_ID (Option 0 of BR_FEATURE_SETTING)
0xDFE4	BR_CLKMODE_SETTING	Initialize CLKMODE (Option 1 of BR_FEATURE_SETTING)
0xDFE7	BR_AUTO_BAUD	Automatically detect UART Baud rate
0xDFEA	BR_UART_BSL	Re-enter UART BSL Mode
0xDFED	BR_FLASH_BACKGROUND_PROGRAM	Program code/data in the background
0xDFF0	BR_FLASH_BACKGROUND_ERASE	Erase flash in the background
0xDFF3	BR_FLASH_BACKGROUND_ERASE_ABORT	Abort Flash Erase (background)
0xDFF6	BR_FLASH_PROGRAM	Program code/data into flash
0xDFF9	BR_FLASH_ERASE	Erase flash

The erase and program user routines can be called from XRAM or Flash. [Table 22-2](#) shows the description of how to use and call these routines.

Boot ROM User Routines

Table 22-2 Calling Flash User Routines

Name	Called from	Target
BR_FLASH_PROGRAM	Flash Bank 0	Flash Bank 0
BR_FLASH_ERASE	XRAM	Flash Bank 0
BR_FLASH_BACKGROUND_PROGRAM	XRAM	Flash Bank 0
BR_FLASH_BACKGROUND_ERASE		
BR_FLASH_BACKGROUND_ERASE_ABORT		

22.1 Flash Bank Read Mode Status Subroutine

The Read Mode status subroutine allows the checking of ready-to-read Mode status of the Flash Bank. This routine is especially useful when user uses **BR_FLASH_BACKGROUND_ERASE_ABORT** user-routine. After calling the erase-abort routine, user can call this Read Mode Status user-routine to check if erase has been successfully aborted and the Flash is in Read Mode already.

Before calling this subroutine, the user must ensure that the input R7 is configured to the selected Flash Bank. The carry flag is clear when Flash is in Read Mode, otherwise it is set. For wrong input, the carry flag is set.

Table 22-3 Specifications of Flash Bank Read Mode Status subroutine

Subroutine	BR_FLASH_READ_MODE_STATUS
Input	R7 of current Register Bank: Selected Flash Bank 00 _H Flash Bank 0 Others: Reserved (Invalid options)
Output	C = 0:Selected Flash Bank is in Read Mode C = 1:Selected Flash Bank is not in Read Mode C = 1:Invalid option is selected
Stack size required	6
Resource used/destroyed	A

Boot ROM User Routines

22.2 Get 4 Bytes Information

This routine allow the tool chain software or even user code to read out the Chip Identification Number (see [Chapter 1.4](#)) or User Identification Number (USER_ID; see [Chapter 5.1](#)) for device.

Table 22-4 Specifications of BR_GET_4_BYTES_INFO subroutine

Subroutine	BR_GET_4_BYTES_INFO
Input	R7 of current Register Bank: Option 00 _H Option 0 Chip Identification Number 01 _H Option 1 User Identification Number (USER_ID) Others: Reserved (Invalid options)
	R5 of current Register Bank: Pointer for the 4-byte IRAM buffer
Output	The IRAM buffer is filled with the following for Option 0: [R5]: Chip Identification Number (MSB) [R5+1]:Chip Identification Number [R5+2]:Chip Identification Number [R5+3]:Chip Identification Number (LSB)
	The IRAM buffer is filled with the following for Option 1: [R5]: USER_ID (MSB) [R5+1]:USER_ID [R5+2]:USER_ID [R5+3]:USER_ID (LSB)
	The IRAM buffer is filled with the following for invalid option: [R5]: 0x00 [R5+1]:0x00 [R5+2]:0x00 [R5+3]:0x00
	C = 0; Fetch is successful C = 1; Fetch is not successful (Option is invalid)
Stack size required	8
Resource used/destroyed	A
	R1 and R7 of current Register Bank (2 bytes)

22.3 Feature Setting Subroutine

Feature Setting subroutine is provided in the Boot ROM to initialize some settings. Currently only 2 options provided, CLKMODE setting (BR_CLKMODE_SETTING) and programming of USER_ID (BR_PROG_USER_ID). See

The CLKMODE setting sets the frequency to 8MHz or 24MHz while USER_ID programming programs the 4 bytes user identification (USER_ID) information into device. These 4 bytes of User Identification Number will consists of BMI, BMI (to determine entry of Mode) and some initialization like clock frequency setting and enabling peripherals. This BR_PROG_USER_ID routine will issue a soft reset once programming is completed. Thus there is no successful indication, except a reset will occur and device will bootup in the programmed boot-mode. This BR_PROG_USER_ID routine will also check if NMISR is 0x00. If yes, it will clear NMICON and EA and continue programming USER_ID. User should take care of their watchdog service routine and to disable the wdt before calling this routine as there is no return back after calling this routine.

Table 22-5 Specifications of CLKMODE Setting subroutine

Subroutine	BR_CLKMODE_SETTING (BR_FEATURE_SETTING - Option 0)
Input	R7 of current Register Bank: Option 00 _H Option 0 - BR_CLKMODE_SETTING Others: Reserved (Invalid options)
	R5 of current Register Bank: 00 - 8MHz 80 - 24MHz Others - Reserved
Output	C = 0:Correct option is selected, CLK Mode is set accordingly. C = 1:Invalid option is selected
Stack size required	9
Resource used/destroyed	A

Boot ROM User Routines

Table 22-6 Specifications of Program USER_ID subroutine

Subroutine	BR_PROG_USER_ID (BR_FEATURE_SETTING - Option 1)
Input	R7 of current Register Bank: Option 01 _H Option 1-BR_PROG_USER_ID Others: Reserved (Invalid options) R5 of current Register Bank: IRAM start address for 4-byte User Identification Number [R5]: USER_ID_3 [R5+1]:USER_ID_2 [R5+2]:USER_ID_1 [R5+3]:USER_ID_0 4-byte User ID Information (in IRAM) SFR NMISR = 00 _H Stack Pointer (SP) Setting: 0x07 <= SP <= 0x60 or 0xC0 <= SP <= 0xE0
Output¹⁾	C = 1:Programming of User ID has failed C = 1:Routine is exited as NMISR is not 00 _H - error
Stack size required	10
Resource used/destroyed	DPTR, A R0, R1, R3, R5, R6 and R7 of current Register Bank (6 bytes) IRAM Address 0x80 - 0xBF (64 bytes)

1) Routine is exited and output is observed only when programming of USER_ID has failed. When successful programming of USER_ID, a soft reset will be triggered and it will boot-up in the programmed Boot-up Mode depending on BMI.

22.4 UART Auto Baud Subroutine

This routine allows application software to configure UART settings upon reception of a synchronization byte from the host. User application would block, once this subroutine is called, throughout reception of synchronization byte, baudrate calculation and transmission of response byte. The protocol of the auto baud is described in [Figure 6-1](#) while the port pins used in the auto baud are defined in [Table 6-1](#)

User has to take care of the clock frequency used and program the BCON.BGSEL SFR field bit according to their desired auto baud rate.

Boot ROM User Routines

Note: Clock Frequency and BGSEL SFR field bit are not modified in this routine. These field bits can be modified by user first before calling this user-routine for desired baud rate and clock frequency

Note: SPD setting is not disabled upon entering this user-routine. User must ensure that there are no conflicts in the port pins used by SPD and auto baud.

Note: It is highly recommended that all interrupts should be disabled before calling this routine as not to interfere with the auto baud detection and calculation.

Table 22-7 Specifications of BR_AUTO_BAUD subroutine

Subroutine	BR_AUTO_BAUD
Input	BCON.BGSEL SFR field bit to be set accordingly
Output	This subroutine configures port pins (to input/output, RXD/TXD) and baud rate for UART based on autobaud value of 005555 _H , or 0055AA _H received. It subsequently transmits the response character, 55 _H (acknowledge code) if initialization is completed successfully.
Stack size required	11
Resource used/destroyed	Port related SFRs, A, SBUF, SCON, MODPISEL1, MODPISEL2, TCON, TMOD, TH0, TL0, TR0, BCON, LINST, T2_T2CON, T2_RC2H/L, T2_T2MOD, T2_T2H/L IRAM address 0x7F

22.5 UART BSL Routine

This routine allows application software to jump back to UART BSL Mode for various modes like programming, erasing of XRAM, FLASH..etc. Auto baud detection will be done in this routine. User has to take care of the clock frequency used and program the BCON.BGSEL SFR field bit according to their desired auto baud rate. For details of the feature of UART_BSL Mode, refer to Boot-loader chapter.

Note: Clock Frequency and BGSEL SFR field bit are not modified in this routine. These field bits can be modified by user first before jumping back to this user-routine for desired baud rate and clock frequency

Note: This routine will NOT return as it will be always be waiting for any uart bsl header, therefore an LJMP to this routine should be used (instead of LCALL).

Note: SPD setting is disabled upon entering this user-routine. And once SPD is disabled, it cannot be enabled again, unless via reset.

Table 22-8 Specifications of BR_UART_BSL subroutine

Subroutine	BR_UART_BSL
Input	IEN0.EA = 0
	NMICON = 0x00
	BCON.BGSEL SFR field bit to be set accordingly
	Stack Pointer (SP) Setting: 0x07 <= SP <= 0x60 or SP = 0xE0
Output	-
Stack size required	-
Resource used/destroyed	-

22.6 Flash Program Subroutine

Each call of the Flash program subroutine allows the programming of:

- 32 of data bytes (WL aligned) into selected Flash wordline.

Before calling this subroutine, the user must ensure that the Flash content to be programmed should be buffered in an identically-sized IRAM buffer. Two types of Flash programming routines will be offered to users. The calling of the erase and program user routines can be called from XRAM or Flash. [Table 22-2](#) shows the description of how to use and call these routines.

The first type of routine (supports non-background programming), [BR_FLASH_PROGRAM](#), will wait until programming is done before allowing user program to continue with its execution. This type of routine is necessary for users who need to program the Flash bank where the user code is in execution, especially when there will be only one flash bank. The Flash cannot be in both program mode and read mode at the same time. It is also useful for users who wish to program the Flash Bank where the interrupt vectors are defined as interrupts cannot be handled when the Flash Bank is in program mode as the interrupt handler cannot be fetched. For Type 1 program routine, Boot ROM will control the code, the FNMIFLASH flag is cleared automatically and therefore need not be handled by users.

Table 22-9 Specifications of FLASH_PROGRAM subroutine

Subroutine	BR_FLASH_PROGRAM
-------------------	------------------

Boot ROM User Routines

Table 22-9 Specifications of FLASH_PROGRAM subroutine (cont'd)

Input	Flash WL aligned address to be programmed R6 of current Register Bank: DPH R7 of current Register Bank: DPL ¹⁾
	R5 of current Register Bank: IRAM start address for 32-byte Flash data
	32-byte Flash data
	All interrupts including NMI must be disabled (IEN0.EA=0, NMICON=00 _H)
	SFR NMISR = 00 _H
Output	PSW.CY: 0 = Flash programming is successful 1 = Flash programming is not successful
	DPTR is incremented by 20 _H ²⁾
Stack size required	12
Resource used/destroyed	DPTR, A
	R0, R2, R6 and R7 of current Register Bank (4 bytes)

1) The last 5 LSB of the DPL is 0 for an aligned WL address, for e.g. 00H, 20H, 40H, 60H, 80H, A0H, C0H and E0H.

2) DPTR is only incremented by 20_H when PSW.CY is 0

The second type of routine (supports background programming), **BR_FLASH_BACKGROUND_PROGRAM**, allows the user program code to continue execution after the programming routine is called. User will wait until the Flash NMI event is generated; bit FNMIFLASH in register NMISR is set, and if enabled via NMIFLASH, an NMI to the CPU is triggered to enter the Flash NMI service routine. At this point, the Flash bank is in ready-to-read mode i.e. programming is done.

Table 22-10 Specifications of FLASH_BACKGROUND_PROGRAM subroutine

Subroutine	BR_FLASH_BACKGROUND_PROGRAM
-------------------	-----------------------------

Boot ROM User Routines

Table 22-10 Specifications of FLASH_BACKGROUND_PROGRAM subroutine

Input	Flash WL aligned address to be programmed: R6 of current Register Bank: DPH R7 of current Register Bank: DPL ¹⁾
	R5 of current Register Bank: IRAM start address for 32-byte Flash data
	32-byte Flash data
	Flash NMI (NMICON.NMIFLASH) is enabled (1) or disabled (0)
	SFR NMISR = 00 _H
Output	PSW.CY: 0 = Flash programming is in progress 1 = Flash programming is not successful
	DPTR is incremented by 20H ²⁾
Stack size required	8
Resource used/destroyed	DPTR, A
	R0, R2, R6 and R7 of current Register Bank (4 bytes)

1) The last 5 LSB of the DPL is 0 for an aligned WL address, for e.g. 00H, 20H, 40H, 60H, 80H, A0H, C0H and E0H.

2) DPTR is only incremented by 20_H when PSW.CY is 0

22.7 Flash Erase Subroutine

Each call of the Flash erase subroutine allows sector(s) erase of the selected Flash Bank. At any one time, more than one Flash Sectors can be selected for erase.

Before calling this subroutine, the user must ensure that inputs are set accordingly. Two types of Flash erasing routines will be offered to users. The calling of the erase and program user routines can be called from XRAM or Flash. [Table 22-2](#) shows the description of how to use and call these routines.

The first type of routine (supports non-background erasing), [BR_FLASH_ERASE](#), will wait until erasing is done before allowing user program to continue with its execution. This routine will be aborted if the FNMIVDDP, FNMIVDD or FNMIPLL flag is set while they are being polled for error. This type of routine is necessary for users who need to erase the Flash bank where the user code is in execution, especially when there is only one Flash Bank. The Flash cannot be in both erase mode and read mode at the same time. It is also useful for users who wish to erase the Flash Bank where the interrupt vectors are defined as interrupts cannot be handled when the Flash is in erase mode as the interrupt handler cannot be fetched. For Type 1 erase routine, Boot ROM will control the code, the FNMIFLASH flag is cleared automatically and therefore need not be handled by users.

Boot ROM User Routines

Note: As program will return to user code after erasing is completed, customer should take care that they do not erase their user program code as well.

Note: When invalid/No input(s) is provided for this routine, C flag will be set, and routine will be exited. Nothing else will be done.

Table 22-11 Specifications of FLASH_ERASE subroutine

Subroutine	BR_FLASH_ERASE
Input¹⁾	R7 of Current Register Bank: Select sector(s) to be erased for the Flash Bank 0 LSB represents sector 0, MSB represents sector 7. R6 of Current Register Bank: Select sector(s) to be erased for the Flash Bank 0 LSB represents sector 8 and bit 1 represents sector 9. SFR NMISR = 00 _H All interrupts including NMI must be disabled (IEN0.EA=0, NMICON=00 _H)
Output	PSW.CY: 0 = Flash erasing is successful 1 = Flash erasing is not successful 1 = Invalid/No input(s) is given
Stack size required	12
Resource used/destroyed	DPTR, A R2 and R7 of current Register Bank (2 bytes)

1) The inputs should be set as 0 if the sector(s) of the bank is/are not to be selected for erasing.

The second type of routine (supports background erasing), **BR_FLASH_BACKGROUND_ERASE**, allows the user program code to continue execution after the erasing routine is called. User will wait until the Flash NMI event is generated; bit FNMIFLASH in register NMISR is set, and if enabled via NMIFLASH, an NMI to the CPU is triggered to enter the Flash NMI service routine. At this point, the Flash bank is in ready-to-read mode i.e. erasing is done.

Note: When invalid/No input(s) is provided for this routine, C flag will be set, and routine will be exited. Nothing else will be done.

Table 22-12 Specifications of FLASH_BACKGROUND_ERASE subroutine

Subroutine	BR_FLASH_BACKGROUND_ERASE
Input¹⁾	R7 of Current Register Bank: Select sector(s) to be erased for the Flash Bank 0 LSB represents sector 0, MSB represents sector 7. R6 of Current Register Bank: Select sector(s) to be erased for the Flash Bank 0 LSB represents sector 8 and bit 1 represents sector 9. SFR NMISR = 00 _H Flash NMI (NMICON.NMIFLASH) is enabled (1) or disabled (0)
Output	PSW.CY: 0 = Flash erasing is in progress 1 = Flash erasing is not successful 1 = Invalid/No input(s) is given
Stack size required	8
Resource used/destroyed	DPTR, A R2 of current Register Bank (1 bytes)

1) The inputs should be set as 0 if the sector(s) of the bank is/are not to be selected for erasing.

22.8 Abort Flash Erase Subroutine

Each complete erase operation on a Flash bank requires approximately 100 ms, during which read and program operations on the Flash bank cannot be performed. For the XC82x, provision has been made to allow an on-going erase operation (type 2, [BR_FLASH_BACKGROUND_ERASE](#)) to be interrupted so that higher priority tasks such as reading/programming of critical data from/to the Flash bank can be performed. Hence, erase operations on selected Flash bank sector(s) may be aborted to allow data in other sectors to be read or programmed. To minimize the effect of aborted erase on the Flash data retention/cycling and to guarantee data reliability, the following points must be noted for each Flash bank:

- An erase operation cannot be aborted earlier than 5 ms after it starts.
- Maximum of two consecutive aborted erase (without complete erase in-between) are allowed on each sector.
- Complete erase operation (approximately 100 ms) is required and initiated by user-program after a single or two consecutive aborted erase as data in relevant sector(s) is corrupted.
- For the specified cycling time, each aborted erase constitutes one program/erase cycling.

Boot ROM User Routines

- Maximum allowable number of aborted erase for each D-Flash sector during lifetime is 2500.

The Flash erase abort subroutine call cannot be performed anytime within 5 ms after the erase operation has started. This is a strict requirement that must be ensured by the user. Otherwise, the erase operation cannot be aborted. Once exited from this routine, user can call **BR_FLASH_READ_MODE_STATUS** user routine to check if the abort erase operation has been completed. When the selected bank is already in Read Mode, it indicates that the abort erase operation has been completed.

Table 22-13 Specifications of Flash Erase Abort subroutine

Subroutine	BR_FLASH_BACKGROUND_ERASE_ABORT
Input	Flash Bank is in erase mode
Output	C = 0: Flash erase abort is in progress C = 1: Flash erase abort is not started
Stack size required	3
Resource used/destroyed	A

23 ROM Library

On top of the User routines, a set of useful ROM library routines is also provided to user to be used in user application. The ROM Library provided are:

- **Fixed Point ROM Library**,
- **LED and Touch-Sense Controller ROM Library**,
- **MDU ROM Library (MATH Function)**,
- **EEPROM Emulation ROM Library**

User application would be able to call these routines by calling the functions provided in **Table 23-1** for XC82x device. Details of the ROM library are covered in its respective section.

Table 23-1 XC82x ROM Library function and its Address

Addr	Name	Description
0xD649	_PI_CONTROLLER_G16	FIXEDPOINT ROM Library
0xD64F	_PI_CONTROLLER_G256	FIXEDPOINT ROM Library
0xD655	_P_CONTROLLER_G16	FIXEDPOINT ROM Library
0xD65B	_P_CONTROLLER_G256	FIXEDPOINT ROM Library
0xD75B	_PT1_24	FIXEDPOINT ROM Library
0xD78A	_PT1_32	FIXEDPOINT ROM Library
0xD7CE	_CLARKE	FIXEDPOINT ROM Library
0xD802	SET_CCU6_SECA ¹⁾	FIXEDPOINT ROM Library
0xD812	SET_CCU6_SECB ¹⁾	FIXEDPOINT ROM Library
0xD822	SET_CCU6_SECC ¹⁾	FIXEDPOINT ROM Library
0xD832	SET_CCU6_SECD ¹⁾	FIXEDPOINT ROM Library
0xD842	SET_CCU6_SECE ¹⁾	FIXEDPOINT ROM Library
0xD852	SET_CCU6_SECF ¹⁾	FIXEDPOINT ROM Library
0xDFC0	?C?IMUL_XC	MDU ROM Library
0xDFC3	?C?LMUL_XC	MDU ROM Library
0xDFC6	?C?UIDIV_XC	MDU ROM Library
0xDFC9	?C?ULDIV_XC	MDU ROM Library
0xDFCC	FINDTOUCHEDPAD	LEDTS ROM Library
0xDFCF	SET_LDLINE_CMP	LEDTS ROM Library
0xDFD2	INITEEPROM	EEPROM Emulation ROM Library

ROM Library
Table 23-1 XC82x ROM Library function and its Address (cont'd)

Addr	Name	Description
0xDFD5	FIXEEPROM	EEPROM Emulation ROM Library
0xDFD8	READEEPROM	EEPROM Emulation ROM Library
0xDFDB	WRITEEEPROM	EEPROM Emulation ROM Library

1) These functions are useful for space vector modulation, as these functions can be used as move accelerator i.e. executing without waitstate.

23.1 Fixed Point ROM Library

The Fixed Point Library contains a list of routines that reside in ROM that are accessible to users. The definitions and specifications of the library routines are explained in the following sections.

“Execution Cycles” in the tables does not include the calling instruction “LCALL” which requires 10 clock cycles if it is executed from FLASH or 8 clock cycles if it is executed from XRAM.

23.1.1 P Controller Routine

The P controller routine is the implementation of a proportional control algorithm defined as:

(23.1)

$$Y(k) = G \times K_p \times X(k)$$

where

- Y(k): P controller output
- G: Gain factor of 16 or 256
- K_p: Proportional gain
- X: Error = Reference value - Actual value
- k: Time or instantaneous time

Table 23-2 Specifications of P Controller Routine

Subroutine	_P_controller_G16 or _P_controller_G256
Routine Address	0xD655: _P_controller_G16 0xD65B: _P_controller_G256
C-Prototype	int P_controller_G16(int Ref_val, int Actual_val, char idata *Kp) int P_controller_G256(int Ref_val, int Actual_val, char idata *Kp)
Input	R7, R6 (MSB), Ref_val = 16 bit Reference value R5, R4 (MSB), Actual_val = 16 bit Actual value R3 = idata pointer to Kp_H
Output	R7, R6 (MSB) = 16 bit Y(k) value
Execution Cycle	_P_controller_G16: 138 cclks _P_controller_G256: 112 cclks
Resource used/destroyed	PSW, A, MDU R0, R4, R5 of current Register Bank

Code Example:

```
//global variables
data int Speed;
data int Speed_ref;
data int Speed_kp;
data int P_Speed;

//program
Speed = 16288
Speed_ref = 4000;
Speed_kp = 4096;
P_Speed = P_controller_G16(Speed_ref, Speed, (char idata *)
&Speed_kp);
//P_speed = 0x7F40
```

23.1.2 PI Controller Routine

The PI controller routine is the implementation of a proportional-integral control algorithm defined as:

$$Y(k) = G \times K_p \times X(k) + Y(k-1) + K_i \times X(k) \quad (23.2)$$

where

- Y(k): PI controller output
- Y(k-1): Previous PI controller output
- G: Gain factor of 16 or 256
- Kp: Proportional gain
- Ki: Integral gain
- X: Error = Reference value - Actual value
- k: Time or instantaneous time

Table 23-3 Specifications of PI Controller Routine

Subroutine	_PI_controller_G16 or _PI_controller_G256
Routine Address	0xD649: _PI_controller_G16 0xD64F: _PI_controller_G256
C-Prototype	int PI_controller_G16(int Ref_val, int Actual_val, PI_param *pPI) int PI_controller_G256(int Ref_val, int Actual_val, PI_param *pPI)

Table 23-3 Specifications of PI Controller Routine (cont'd)

Input	R7, R6 (MSB), Ref_val = 16 bit Reference value
	R5, R4 (MSB), Actual_val = 16 bit Actual value
	R3 = idata pointer to Y(k-1)_H
Remark	Required structure: Struct{ Y(k-1)[HLh] Kp[HL] Ki[HL] PI_SAT_HH //high byte of 24 bit saturation value 0xHHFFFF }
Output	R7, R6 (MSB) = 16 bit Y(k) value
Execution Cycle	_PI_controller_G16: 248 cclks _PI_controller_G256: 226 cclks
Resource used/destroyed	PSW, A, MDU
	R0, R2, R4, R5 of current Register Bank

Code Example:

```
//define structure
typedef struct pi_param{
    char        yn[3];
    int         kp;
    int         ki;
    unsigned char PI_SAT_HH;
}PI_param;

//global variables
data int Speed;
data int Speed_ref;
data int PI_Speed;
idata PI_param Speed_control;

//program
Speed = 2048;
Speed_ref = 4000;
Speed_control.yn[0] = 0x01;
Speed_control.yn[1] = 0x4C;
Speed_control.yn[2] = 0x97;
Speed_control.kp = 4096;
Speed_control.ki = 32;
```



```
Speed_control.PI_SAT_HH = 0x1F;
PI_Speed = PI_controller_G16(Speed_ref, Speed, &Speed_control);
//PI_Speed = 0x108E
```

23.1.3 PT1_24 Controller Routine

The PT1_24 controller routine is a digital lowpass filter defined as:

(23.3)

$$Y(k) = Y(k-1) + 2^{-Z} \times [X(k) - Y(k-1)]$$

where

- Y(k): PT1_24 controller output
- Y(k-1): Previous PT1_24 controller output
- Z: Division factor of $1/2^Z$
- X(k): 16 bit input
- k: Time or instantaneous time

Table 23-4 Specifications of PT1_24 Controller Routine

Subroutine	_PT1_24
Routine Address	0xD75B: _PT1_24
C-Prototype	int PT1_24(int X, char Z, char idata *pY)
Input	R7, R6 (MSB) = 16 bit X value R5 = 8 bit Z value, number of right shifts ($1/2^Z$) R3 = idata pointer to Y(k-1)_H
Remark	Y(k-1) = 24 bits [HLh] X must be within 0xC000 to 0x3FFF
Output	R7, R6 (MSB) = 16 bit Y(k) value
Execution Cycle	76 cclk
Resource used/destroyed	PSW, A, MDU R0 of current Register Bank

Code Example:

```
//declare variables
data char Speed[3];
data int result;
data int angle_new;
```

```
data int  angle_old;

//program
Speed[0] = 0xF8;
Speed[1] = 0xF3;
Speed[2] = 0xCB;
angle_new = 0x1800;
angle_old = 0x0800;
result = PT1_24(angle_new - angle_old, 5, &speed);
//result = 0xF9AC;
```

23.1.4 PT1_32 Controller Routine

The PT1_32 controller routine is an integrator defined as:

(23.4)

$$Y(k) = Y(k-1) + Z1 \times X(k) - Z2 \times Y(k-1)$$

where

- Y(k): PT1_24 controller output
- Y(k-1): Previous PT1_24 controller output
- Z: Division factor of $1/2^Z$
- X(k): 16 bit input
- k: Time or instantaneous time

Table 23-5 Specifications of PT1_32 Controller Routine

Subroutine	_PT1_32
Routine Address	0xD78A: _PT1_32
C-Prototype	int PT1_32(int X, int Z2, int idata *pY)
Input	R7, R6 (MSB) = 16 bit X value
	R5, R4 (MSB) = 16bit Z2 value
	R3 = idata pointer to Y(k-1)_H
	MDU_MD4 = Z1_L MDU_MD5 = Z1_H
Remark	Y(k-1) = 32 bits [HLh]
	Z1 must be within 0x0000 to 0x3FFF or X must be within 0xC000 to 0x3FFF
Output	R7, R6 (MSB) = 16 bit Y(k) value
Execution Cycle	118 cclk

Table 23-5 Specifications of PT1_32 Controller Routine (cont'd)

Resource used/destroyed	PSW, A, MDU
	R0 of current Register Bank

Code Example:

```
//global variables
data int    result;
data int    Integrator[2];

//program
Integrator[0] = 0xFF12;
Integrator[1] = 0xE828;
MDU_MD4 = 0x1F; //Low byte Z1
MDU_MD5 = 0x00; //High byte Z1
result = PT1_32(X, 0x109, &Integrator);
//result = 0xFF15
```

23.1.5 Clarke Transform Routine

This routine is used for the transformation of a three phase system into a two phase orthogonal system. It's implementation is defined as:

$$I_{\alpha} = \frac{I_{\text{phaseA}}}{2} \quad (23.5)$$

$$I_{\beta} = \left(\frac{\text{PhaseA} + 2 \times \text{PhaseB}}{\sqrt{3} \times 2} \right) \quad (23.6)$$

where

- Y(k): PT1_24 controller output
- Y(k-1): Previous PT1_24 controller output
- Z: Division factor of $1/2^Z$
- X(k): 16 bit input
- k: Time or instantaneous time

Table 23-6 Specifications of Clarke Transform Routine

Subroutine	_Clarke
Routine Address	0xD7CE: _Clarke

Table 23-6 Specifications of Clarke Transform Routine

C-Prototype	void Clarke(int idata *I_phaseAB, int idata *I_alphabeta)
Input	R7 = idata pointer to I_phaseA_H
	R5 = idata pointer to I_alpha_H
Remark	char idata *I_phaseAB Expected address arrangement in IRAM: I_phaseA_H (MSB), I_phaseA_L, I_phaseB_H, I_phaseB_L
	char idata *I_alphabeta Expected address arrangement in IRAM: I_alpha_H (MSB), I_alpha_L, I_beta_H, I_beta_L
Output	Result of I_alpha, I_beta saved in address location of 2 nd input parameter
Execution Cycle	84 cclk
Resource used/destroyed	PSW, A, MDU
	R0, R1 of current Register Bank

Code Example:

```
//declare variables
data int   I_phaseA   _at_ (0x36);
data int   I_phaseB   _at_ (0x38);
data int   I_alpha    _at_ (0x4A);
data int   I_beta     _at_ (0x4C);

//program
I_phaseA = -2048;
I_phaseB = 8192;
Clarke(&I_phaseA, &I_alpha);
//I_alpha = 0xFC00, I_beta = 0x102A
```

23.2 LED and Touch-Sense Controller ROM Library

The LEDTS ROM Library contains two routines that reside in ROM that are accessible to users. The definitions and specifications of the library routines are organized into the following major sections.

- [SET_LDLINE_CMP Function \(LED and TS\)](#)
- [FINDTOUCHEDPAD Function \(TS\)](#)
- [Use of the functions in Interrupts](#)

The call functions for LEDTS are listed in [Table 23-7](#).

Table 23-7 LEDTS ROM Library Routine Table

Address	Name	Description
0xD0CF	SET_LDLINE_CMP	Program SFRs LTS_LDLINE and LTS_COMPARE (LED enabled only).
0xD0CF	SET_LDLINE_CMP	Program SFRs LTS_LDLINE and LTS_COMPARE (TS enabled only).
0xD0CF	SET_LDLINE_CMP	Program SFRs LTS_LDLINE and LTS_COMPARE (LED and TS are enabled)
0xD0CC	FINDTOUCHEDPAD	Get Average, calculate LowTrip and assess whether pad(s) is being touched or not.

These routines are called from Time Slice or Time Frame Interrupts. This is illustrated in [Section 23.2.3](#).

23.2.1 SET_LDLINE_CMP Function (LED and TS)

This function programs sfrs LTS_LDLINE and LTS_COMPARE (brightness/oscillation window) for LED and/or Touch-sense, based on the users' input parameters. The function takes sfr bit FNCOL value direct from the sfr LTS_GLOBCTL1 to synchronize which data is to be programmed into the sfr LTS_LDLINE for implementing an LED display¹⁾. Likewise, the function also updates the sfr LTS_COMPARE for individual LED or Touch-sense PADTurn number. [Figure 23-1](#) gives an overview of the usage of this function while [Figure 23-2](#) shows the respective SFR settings used for this function.

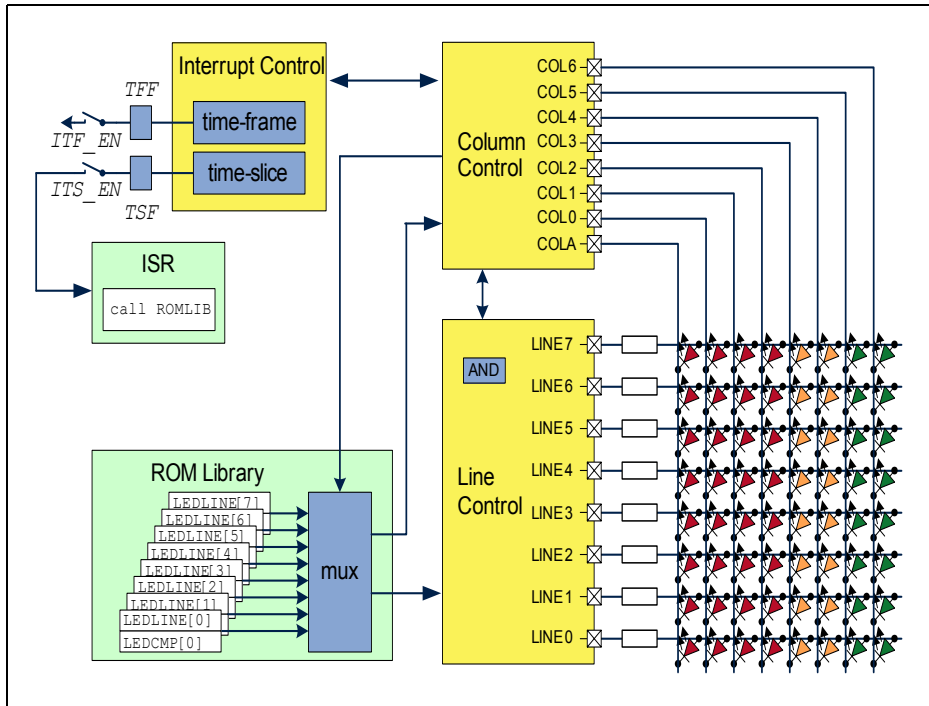


Figure 23-1 SET_LDLINE_CMP Function Overview

This function can be called at every Time Slice or Time Frame interrupt, depending on whether Touch-sense or/and LED is/are enabled. Details of how this functions can be called are shown in [Section 23.2.3](#).

The inputs are shown in [Section 23.2.1.1](#), [Section 23.2.1.2](#) and [Section 23.2.1.3](#) for LED enabled only, TS enabled only and both LED and TS enabled respectively.

1) For Touch-sense, LTS_LDLINE parameter can be written as 0xFF.

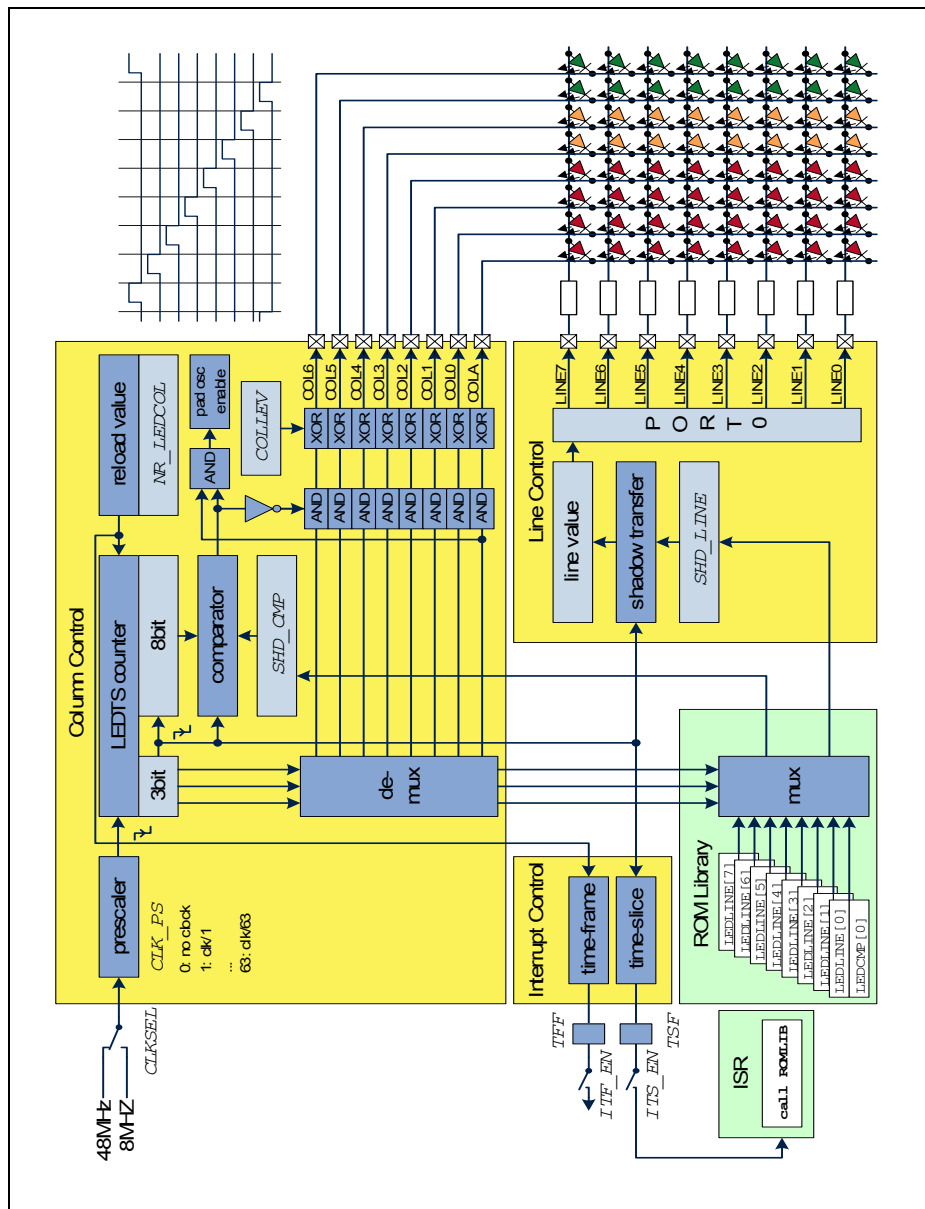


Figure 23-2 SET_LDLINE_CMP Function - SFR Settings

23.2.1.1 Inputs for SET_LDLINE_CMP Function (LED only)

If only LED module is enabled, the inputs overview is shown in [Table 23-8](#). Examples of input parameters for 2, 4, 6 or 8 LEDs enabled are also shown in [Figure 23-3](#).

Table 23-8 Specifications of Setting LDLINE & COMPARE (LED only)

Subroutine	SET_LDLINE_CMP
Address	DFCF _H
Input	<p>R7 of current Register Bank: Start IRAM address of LDLINE parameters</p> <p>R5 of current Register Bank: Start IRAM address of COMPARE parameters</p> <p>LDLINE parameters programmed into IRAM R7, R7+1..etc¹⁾. @R7 = LDLINE parameter for COLA @R7+1 = LDLINE parameter for COL0 ... @R7+(y-1) = LDLINE parameter for COL(y-2)</p> <p>COMPARE parameters programmed into IRAM R5, R5+1..etc¹⁾ @R5 = COMPARE parameter for COLA @R5+1 = COMPARE parameter for COL0 ... @R5+(y-1) = COMPARE parameter for COL(y-2)</p>
Output	<p>Sfr LTS_LDLINE is programmed.</p> <p>Sfr LTS_COMPARE is programmed.</p>
Stack size required	2
Resource used/destroyed	A, R0, R1, R2, R3, R4

1) Depending how many LED(s) is/are enabled; y = no. of LED(s) enabled

ROM Library

Eg. 2 LED + No TS enabled		Eg. 4 LED + No TS enabled		Eg. 6 LED + No TS enabled		Eg. 8 LED + No TS enabled	
LDLINE(A)	R7	LDLINE(A)	R7	LDLINE(A)	R7	LDLINE(A)	R7
LDLINE(0)	R7 + 1	LDLINE(0)	R7 + 1	LDLINE(0)	R7 + 1	LDLINE(0)	R7 + 1
		LDLINE(1)	R7 + 2	LDLINE(1)	R7 + 2	LDLINE(1)	R7 + 2
COMPARE(A)	R5	LDLINE(2)	R7 + 3	LDLINE(2)	R7 + 3	LDLINE(2)	R7 + 3
COMPARE(0)	R5 + 1			LDLINE(3)	R7 + 4	LDLINE(3)	R7 + 4
		COMPARE(A)	R5	LDLINE(4)	R7 + 5	LDLINE(4)	R7 + 5
		COMPARE(0)	R5 + 1			LDLINE(5)	R7 + 6
		COMPARE(1)	R5 + 2	COMPARE(A)	R5	LDLINE(6)	R7 + 7
		COMPARE(2)	R5 + 3	COMPARE(0)	R5 + 1		
				COMPARE(1)	R5 + 2	COMPARE(A)	R5
				COMPARE(2)	R5 + 3	COMPARE(0)	R5 + 1
				COMPARE(3)	R5 + 4	COMPARE(1)	R5 + 2
				COMPARE(4)	R5 + 5	COMPARE(2)	R5 + 3
						COMPARE(3)	R5 + 4
						COMPARE(4)	R5 + 5
						COMPARE(5)	R5 + 6
						COMPARE(6)	R5 + 7
For LED							
For Touch Sense							

Figure 23-3 Input Parameters Example (LED enabled only)

23.2.1.2 Inputs for SET_LDLINE_CMP Function (Touch-sense only)

If only Touch-sense module is enabled, the inputs overview is shown in [Table 23-9](#), [Table 23-10](#) for common and different compare parameters setting (for oscillation window) respectively. Examples of input parameters are also shown in [Figure 23-4](#).

For flexibility, users can choose common compare parameter for all PADTx or choose different compare parameters for individual PADTx. This is also provided to give better sensitivity for respective pads. If CMP_OPTION (at address R5) is programmed as 0xFF, it means that different compare value is selected. Then the different compare parameters are read, at IRAM address R5+1 onwards, for individual pads. If CMP_OPTION is not programmed as 0xFF, then common compare parameter for all pads is selected, and this value is programmed at address R5 itself.

In preparing compare value for Touch-sense, quantization effect is taken into consideration in the function. The respective compare value will be XRL with 2, 4, 8, 16 etc. and finally updating the final value to sfr LTS_COMPARE. Quantization effect gives better sensitivity and is transparent to users as all are achieved by the function.

Table 23-9 Specifications of Setting LDLINE & Common COMPARE (TS only)

Subroutine	SET_LDLINE_CMP
Address	DFCF _H
Input	R7 of current Register Bank: Start IRAM address of LDLINE parameter
	R5 of current Register Bank: Start IRAM address of CMP_OPTION & COMPARE parameters
	LDLINE parameters programmed into IRAM, address R7. @R7 = LDLINE parameter for COLA (TS)
	COMPARE parameters programmed into IRAM, address R5 @R5 = Common COMPARE parameter ¹⁾ for all PADTx
Output	Sfr LTS_LDLINE is programmed. Sfr LTS_COMPARE is programmed.
Stack size required	2
Resource used/destroyed	A, R0, R1, R2, R3, R4

1) This common Compare parameter cannot be 0xFF value.

Table 23-10 Specifications of Setting LDLINE & Different COMPARE (TS only)

Subroutine	SET_LDLINE_CMP
Address	DFCF _H

Table 23-10 Specifications of Setting LDLINE & Different COMPARE (TS only)

Input	R7 of current Register Bank: Start IRAM address of LDLINE parameter
	R5 of current Register Bank: Start IRAM address of CMP_OPTION & COMPARE parameters
	LDLINE parameters programmed into IRAM, address R7. @R7 = LDLINE parameter for COLA (TS)
	COMPARE parameters programmed into IRAM, R5, R5+1..etc ¹⁾ @R5 = CMP_OPTION (0xFF value) @R5+1 = COMPARE parameter for PADT0 @R5+2 = COMPARE parameter for PADT1 ... @R5+z = COMPARE parameter for PADT(z-1)
Output	Sfr LTS_LDLINE is programmed. Sfr LTS_COMPARE is programmed.
Stack size required	2
Resource used/destroyed	A, R0, R1, R2, R3, R4

1) Depending how many Touch-sense pad turn(s) is/are enabled; z = no. of PADTx enabled

23.2.1.3 Inputs for SET_LDLINE_CMP Function (LED and TS)

If both LED and Touch-sense modules are enabled, the inputs overview is shown in [Table 23-11](#) and [Table 23-12](#) for common and different compare parameters setting (for oscillation window) respectively. Examples of input parameters are also shown in [Figure 23-4](#) and [Figure 23-5](#).

For flexibility, users can choose common compare parameter for all PADTx or choose different compare parameters for individual PADTx. This is also provided to give better sensitivity for respective pads. If CMP_OPTION (at address R5) is programmed as 0xFF, it means that common compare value is selected. Then the common compare parameter is read at IRAM address R5+1. If CMP_OPTION is not programmed as 0xFF, then different compare parameters for individual pads are selected.

In preparing compare value for Touch-sense, quantization effect is taken into consideration in the function. The respective compare value will be XRL with 2, 4, 8, 16 etc. and finally updating the final value to sfr LTS_COMPARE. Quantization effect gives better sensitivity and is transparent to users as all are achieved by the function.

Table 23-11 Specifications of Setting LDLINE & Common COMPARE (LEDTS)

Subroutine	SET_LDLINE_CMP
Address	DFCF _H
Input	R7 of current Register Bank: Start IRAM address of LDLINE parameters
	R5 of current Register Bank: Start IRAM address of COMPARE parameters
	LDLINE parameters programmed into IRAM R7, R7+1..etc ¹⁾ . @R7 = LDLINE parameter for COLA (for Touch-sense) @R7+1 = LDLINE parameter for COL0 ... @R7+y = LDLINE parameter for COL(y-1)
	COMPARE parameters programmed into IRAM R5, R5+1..etc ¹⁾ @R5 = Common COMPARE parameter ²⁾ for PADTx (COLA) @R5+1 = COMPARE parameter for COL0 ... @R5+7 = COMPARE parameter for COL6
Output	Sfr LTS_LDLINE is programmed. Sfr LTS_COMPARE is programmed.
Stack size required	2
Resource used/destroyed	A, R0, R1, R2, R3, R4

- 1) Depending how many LED(s) is/are enabled; y = no. of LED(s) enabled. Maximum LEDs enabled are 7 when TS is enabled.
- 2) This common Compare parameter cannot be 0xFF value.

Table 23-12 Specifications of Setting LDLINE & Different COMPARE (LEDTS)

Subroutine	SET_LDLINE_CMP
Address	DFCF _H
Input	<p>R7 of current Register Bank: Start IRAM address of LDLINE parameter</p> <p>R5 of current Register Bank: Start IRAM address of CMP_OPTION & COMPARE parameters</p> <p>LDLINE parameters programmed into IRAM, address R7. @R7 = LDLINE parameter for COLA (for Touch -sense) @R7+1 = LDLINE parameter for COL0 ... @R7+y = LDLINE parameter for COL(y-1)</p> <p>COMPARE parameters programmed into IRAM, R5, R5+1..etc¹⁾ @R5 = CMP_OPTION (0xFF value) @R5+1 = COMPARE parameter for COL0 @R5+2 = COMPARE parameter for COL1 ... @R5+y = COMPARE parameter for COL(y-1) @R5+(y+1) = COMPARE parameter for PADT0 @R5+(y+2) = COMPARE parameter for PADT1 ... @R5+(y+z) = COMPARE parameter for PADT(z-1)</p>
Output	<p>Sfr LTS_LDLINE is programmed.</p> <p>Sfr LTS_COMPARE is programmed.</p>
Stack size required	2
Resource used/destroyed	A, R0, R1, R2, R3, R4

- 1) Depending how many LED(s) and Touch-sense pad turn(s) is/are enabled; y = no. of LED(s) enabled, z = no. of PADTx enabled

Eg. No LED + 8 TS PAD enabled TS PADTx has different compare value		Eg. No LED + 8 TS PAD enabled TS PADTx has common compare value		Eg. 2 LED + 2 TS PAD enabled TS PADTx has different compare value		Eg. 2 LED + 2 TS PAD enabled TS PADTx has common compare value	
LDLINE(A)	R7	LDLINE(A)	R7	LDLINE(A)	R7	LDLINE(A)	R7
CMP_OPTION = 0xFF	R5	Common COMPARE(PADTx)	R5	LDLINE(0)	R7 + 1	LDLINE(0)	R7 + 1
COMPARE (PADT0)	R5 + 1			LDLINE(1)	R7 + 2	LDLINE(1)	R7 + 2
COMPARE (PADT1)	R5 + 2			CMP_OPTION = 0xFF	R5	Common COMPARE(PADTx)	R5
COMPARE (PADT2)	R5 + 3			COMPARE (0)	R5 + 1	COMPARE (0)	R5 + 1
COMPARE (PADT3)	R5 + 4			COMPARE (1)	R5 + 2	COMPARE(1)	R5 + 2
COMPARE (PADT4)	R5 + 5			COMPARE (PADT0)	R5 + 3		
COMPARE (PADT5)	R5 + 6			COMPARE (PADT1)	R5 + 4		
COMPARE (PADT6)	R5 + 7						
COMPARE (PADT7)	R5 + 8						
						For LED	
						For Touch Sense	

Figure 23-4 Input Parameters Examples 1(LED & TS enabled)

ROM Library

Eg. 4 LED + 5 TS PAD enabled TS PADTx has different compare value		Eg. 4 LED + 5 TS PAD enabled TS PADTx has common compare value		Eg. 7 LED + 8 TS PAD enabled TS PADTx has different compare value		Eg. 7 LED + 8 TS PAD enabled TS PADTx has common compare value	
LDLINE(A)	R7	LDLINE(A)	R7	LDLINE(A)	R7	LDLINE(A)	R7
LDLINE(0)	R7 + 1	LDLINE(0)	R7 + 1	LDLINE(0)	R7 + 1	LDLINE(0)	R7 + 1
LDLINE(1)	R7 + 2	LDLINE(1)	R7 + 2	LDLINE(1)	R7 + 2	LDLINE(1)	R7 + 2
LDLINE(2)	R7 + 3	LDLINE(2)	R7 + 3	LDLINE(2)	R7 + 3	LDLINE(2)	R7 + 3
LDLINE(3)	R7 + 4	LDLINE(3)	R7 + 4	LDLINE(3)	R7 + 4	LDLINE(3)	R7 + 4
CMP_OPTION = 0xFF	R5	Common COMPARE(PADTx)	R5	LDLINE(4)	R7 + 5	LDLINE(4)	R7 + 5
COMPARE(0)	R5 + 1	COMPARE(0)	R5 + 1	LDLINE(5)	R7 + 6	LDLINE(5)	R7 + 6
COMPARE(1)	R5 + 2	COMPARE(1)	R5 + 2	LDLINE(6)	R7 + 7	LDLINE(6)	R7 + 7
COMPARE(2)	R5 + 3	COMPARE(2)	R5 + 3	CMP_OPTION = 0xFF	R5	Common COMPARE(PADTx)	R5
COMPARE(3)	R5 + 4	COMPARE(3)	R5 + 4	COMPARE(0)	R5 + 1	COMPARE(0)	R5 + 1
COMPARE (PADT0)	R5 + 5			COMPARE(1)	R5 + 2	COMPARE(1)	R5 + 2
COMPARE (PADT1)	R5 + 6			COMPARE(2)	R5 + 3	COMPARE(2)	R5 + 3
COMPARE (PADT2)	R5 + 7			COMPARE(3)	R5 + 4	COMPARE(3)	R5 + 4
COMPARE (PADT3)	R5 + 8			COMPARE(4)	R5 + 5	COMPARE(4)	R5 + 5
COMPARE (PADT4)	R5 + 9			COMPARE(5)	R5 + 6	COMPARE(5)	R5 + 6
				COMPARE(6)	R5 + 7	COMPARE(6)	R5 + 7
				COMPARE (PADT0)	R5 + 8		
				COMPARE (PADT1)	R5 + 9		
				COMPARE (PADT2)	R5 + 10		
				COMPARE (PADT3)	R5 + 11		
				COMPARE (PADT4)	R5 + 12		
				COMPARE (PADT5)	R5 + 13		
				COMPARE (PADT6)	R5 + 14	For LED	
				COMPARE (PADT7)	R5 + 15	For Touch Sense	

Figure 23-5 Input Parameters Examples 2 (LED & TS enabled)

23.2.2 FINDTOUCHEDPAD Function (TS)

The touch-sense concept is based on a pad being a capacitor between the padline and ground. A finger approaching this pad will alter the capacitance. To measure this change, the arrangement is such that together with a resistor and the I/O pad we have an RC oscillator. The oscillations are counted in an 8-bit counter over a pre-determined period, essentially forming a frequency counter. This frequency counting (LTS_TSCTR) happens for each pad line. User can determine the number of samples to accumulate to form a total frequency counter (TOTAL_TSCTRL/H).

The function FINDTOUCHEDPAD, needed for successfully detecting if a pad has been touched, consists the following features:

- Find Average
- Find LowTrip
- Find which pad(s) is/are touched, if any. Generate status flag(s).

Figure 23-6 gives an overview of this function while **Figure 23-7** shows the respective SFR settings used for this function.

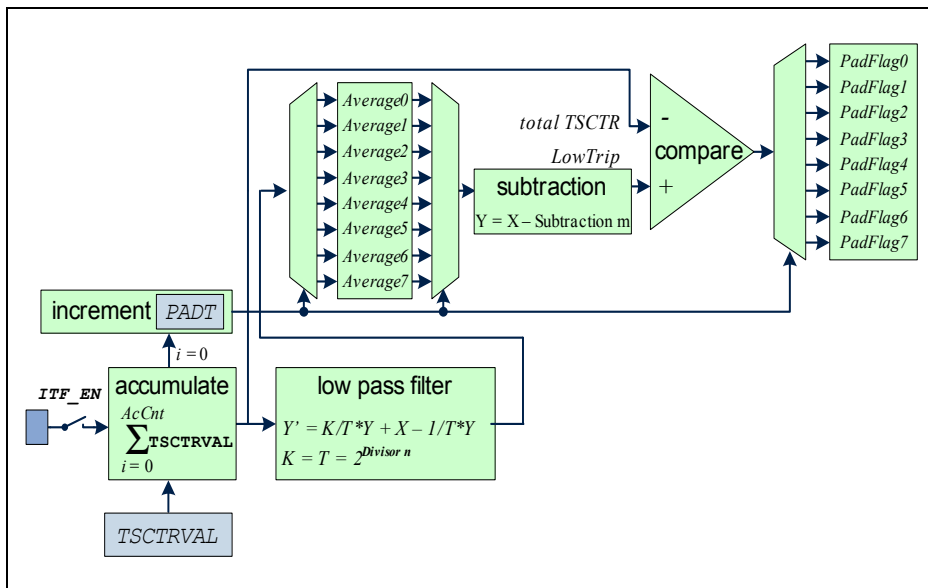


Figure 23-6 FINDTOUCHEDPAD Function Overview

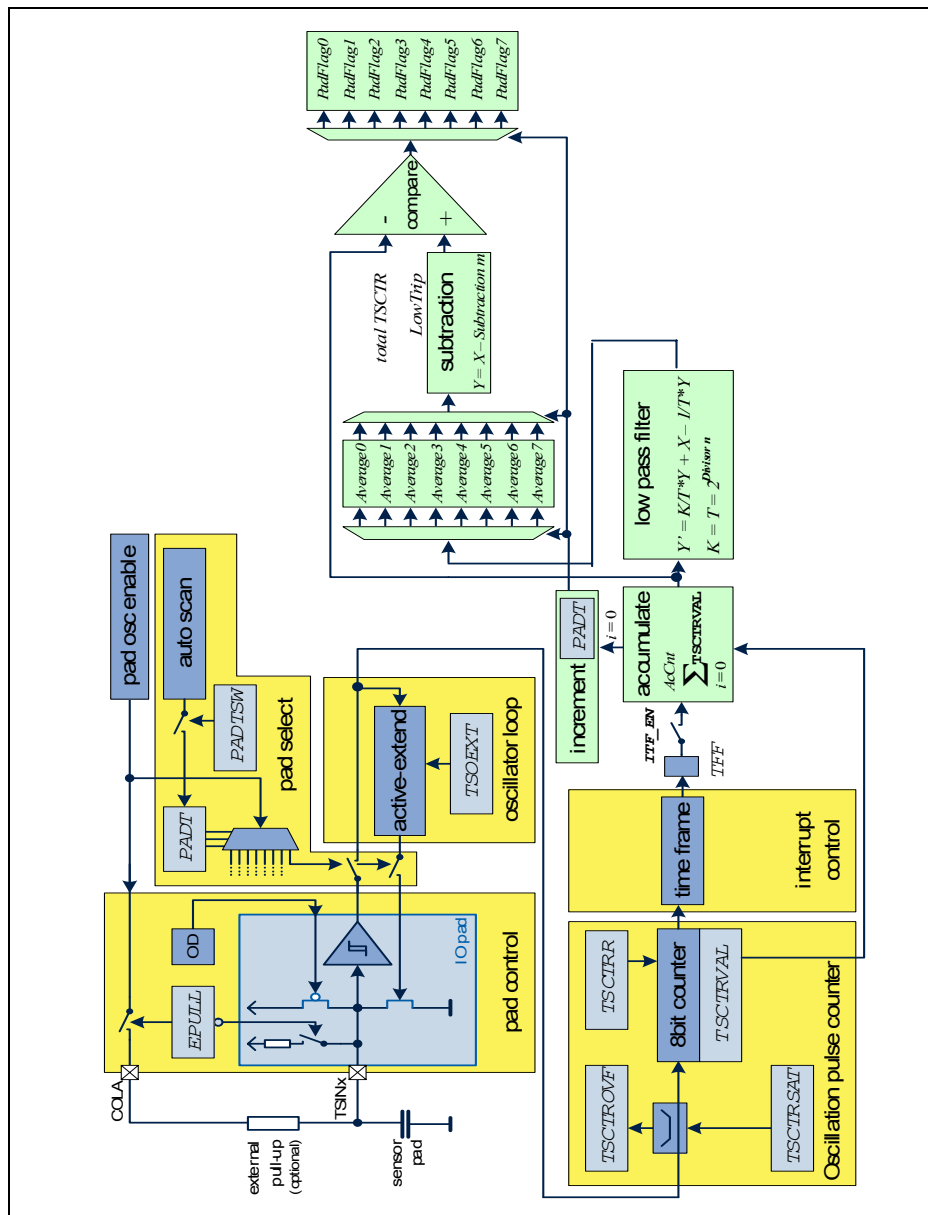


Figure 23-7 FINDTOUCHEDPAD Function - SFR Settings

ROM Library

This function calculates a running average for each Pad Turn to eliminate any spurious peaks and troughs in the pad frequencies and to create a stable value from which the trip points can be calculated. The average is derived from the total values from number of samples (input, AccumulatorCounter) and included low pass filter gain (input, divisor n). It then takes the stored average and calculates a trip point by subtracting a value (input, subtraction m) from the average. This low trip value will be the determining factor if the key has been touched. Status Flags (output, PadResult, PadFlag, PadError) will be the indication key factors of what is/are detected in the function. Finding average and low trip values are skipped when at least one of the pad flags is set (PADFLAG Status!= 0x00).

Common or different Trip point for each PADT can be chosen and is determined by the Subtraction Option at IRAM address 0x32. If Subtraction Option is 0x00, common subtraction m value is initialized at IRAM address 0x34. If different subtraction m values are selected, Subtraction Option is programmed as the start IRAM address of these subtraction m values. Common or different Oscillation window period (LTS_COMPARE) can be determined by users as well, refer to [Section 23.2.1](#).

The details of the subroutine are listed in [Table 23-13](#). The parameters for the function are shown also in [Section 23.2.2.1](#). Implementation details is shown in [Section 23.2.2.2](#).

Table 23-13 Specifications of Find Touched Pad Subroutine

Subroutine	FINDTOUCHEDPAD
Address	DFCC _H
Input	Input Parameters at IRAM address or SFR input setting LTS_GLOBCTL0.TS_EN = 1 ¹⁾ Enable Touch-Sense function control and its respective settings LTS_TSCTL.PADTSW = 0 User needs to disable the hardware control to use this function. ²⁾
IRAM address 0x30	Accumulator Counter The number of samples to be accumulated to calculate the average value. Values supported are 1 to 255
IRAM address 0x31	ShortCount The value to be compared with a counter value (named PadDownCounter (PDC)) to indicate a valid range to set the Result flag. PDC < ShortCount indicates valid range. PDC is initialized to be 0xFF at start of a detected padtouch, and is decremented once at each period.

Table 23-13 Specifications of Find Touched Pad Subroutine (cont'd)

IRAM address 0x32	Subtraction Option or Subtraction m Address³⁾ The option to choose between common subtraction value or different respective subtraction values for all touch pads. If option is 0x00, it means common subtraction is chosen for all touch pads. The common subtraction m value is at Iram address 0x34. If not 0x00, it means different subtraction values are chosen for respective touch pads. In this case, the Iram start address for the subtraction m is given here.
IRAM address 0x33	Divisor n Low pass filter gain (2^n) for calculating the average value where n is 1, 2, 3, 4, 5, 6, 7, 8 for low pass filter gain of 2, 4, 8... 256
IRAM address 0x34	Subtraction m value: Value used to minus from Average to get the LowTrip (Trip point) value.
or user-defined IRAM address in 0x32	Common subtraction m value⁴⁾ or <ul style="list-style-type: none"> Subtraction m Address⁵⁾: Subtraction m value of PADT0 Subtraction m Address+1: Subtraction m value of PADT1 Subtraction m Address+2: Subtraction m value of PADT2 Subtraction m Address+z: Subtraction m value of PADTz where z is no. of PADTx enabled
IRAM address 0x2D	PadError Status If bit is 1, function will be exited for that Touch-sense Pad Turn. No analysis will be done. Users should clear this status when PadFlag bit is 0 for future analysis.
IRAM address 0x2E	PadResult Status If bit is 1, function will be exited for that Touch-sense Pad Turn. No analysis will be done. Users should clear this status when PadFlag bit is 0 for future analysis.
Output	Output Parameters at IRAM address:
IRAM address 0x2D	PadError Status Bit 0 indicates the error status of PADT0 Bit 1 indicates the error status of PADT1 .. Bit 7 indicates the error status of PADT7 This byte/bit(s) indicate(s) that the pad(s) is/are touched for too long. This byte/bit(s) is/are not cleared by function, and must be cleared by user. Access this status only when PadFlag status is 0.

Table 23-13 Specifications of Find Touched Pad Subroutine (cont'd)

IRAM address 0x2E	PadResult Status Bit 0 indicates the result status of PADT0 Bit 1 indicates the result status of PADT1 .. Bit 7 indicates the result flag of PADT7 This byte/bit(s) indicate(s) that the pad(s) is/are touched within valid range. This byte/bit(s) is/are not cleared by function, and must be cleared by user. Access this status only when PadFlag status is 0.
IRAM address 0x2F	PadFlag Status Bit 0 indicates the flag status of PADT0 Bit 1 indicates the flag status of PADT1 .. Bit 7 indicates the flag status of PADT7 When this byte/bit(s) is/are 1, it indicate(s) that the pad(s) is/are identify as being touched and analysis is in progress. When analysis is completed, this byte/bit(s) will be clear by the function. This byte/bit(s) will be set and cleared by function, users must not tamper with this. Users can read the respective PadError and PadResult bit status when respective PadFlag bit status is 0.
Stack size required	2
Resource used/destroyed	A, R0, R1, R3, R4, R5, R6, R7
	IRAM address 0x2D - 0x2F (3 byte)
	XRAM address 0xF0EC - 0xF0FF (max 20 bytes)

- 1) To enable Touch-sense control, as well as number of Touch-Sense pad turns, and the interrupt flag(s) (Time Frame/Time Slice)
- 2) The function will control the padt and configure it in LTS_TSCTL.PADT SFR field.
- 3) Users can programmed this value as 0x34 (for example). And at IRAM address 0x34, the subtraction m value for PADT0, at 0x35, subtraction m value for PADT1, at 0x36, the subtraction m value for PADT2...etc
- 4) To use common subtraction m value for all touch pads, program subtraction option as 0x00 in iram address 0x32.
- 5) Subtraction m address is defined in iram address 0x32

Notes:

- This function uses fixed IRAM address 0x2D to 0x34, and XRAM address 0xF0EC - 0xF0FF. These addressees are not to be destroyed by users at all time. In the event of error detection, users can clear these addressees to start afresh the calculation.
- The number of Touch-Sense Pad Turns enabled must be sequential, e.g. PADT0, PADT1 etc. for the software round robin of the function to work properly. Users

ROM Library

cannot enable 3 PADTx and defined them as PADT0, PADT4 and PADT6 for example, it has to be PADT0, PADT1 and PADT2.

- PADTx will remain for (AccumulatorCounter+1¹⁾) time, and the function will increment the PADTx when TSCTR_COUNTER²⁾ is 0x00 and will repeat back to PADT0 once the highest enabled PADTx is set. For example, if AccumulatorCounter is defined as 0x04, and 2 Touch-sense Pad Turns are enabled, the sequences for PADTx will be PADT1, PADT1, PADT1, PADT1, PADT1, PADT0, PADT0, PADT0, PADT0, PADT0, PADT1..etc. The cycle continues.
- This function is normally called in a Time Frame interrupt. But user can also call this function in a Time Slice interrupt.
- A check whether LTS_GLOBCTL1.FNCOL=0x07 is implemented in this function, therefore users can safely call this function in Time Slice interrupt, after SET_LDLINE_CMP function, without extra code check in between. The function will check if this Time Slice is for Touch-sense before executing the rest of the function. If LTS_GLOBCTL1.FNCOL is not 0x07, function will exit. See [Figure 23-13](#).

1) Extra one count is for calculation of Average value.

2) TSCTR_COUNTER is a parameter in XRAM that holds the user-defined input AccumulatorCounter and will decrement every time FINDTOUCHEDPAD function is entered when LTS_GLOBCTL1.FNCOL = 0x07 (for Touch-sense)

23.2.2.1 Outputs of Function

This function checks if the current counter value is below the trip point and sets a flag in a variable called PadFlag Status. Once a PadFlag Status has been set, a software implemented Pad Down Counter (PDC) starts to decrement each time the function is called provided the PadFlag is set. When PadFlag Status is set, it means the particular pad is being assessed and result is either valid touch, short touch (invalid) or long touch (error).

Once the pad down counter value reaches above the trip point, the PDC is checked and compared with a pre-determined threshold value, ShortCount. If above this threshold, the PadFlag is simply cleared and no further action takes place. This is regarded as a invalid key press.

If the PDC is below the threshold but >0, the PadFlag Status is cleared and the corresponding bit in a variable called PadResult Status is set. This means that the padtouch is being recognized. But once the PDC reaches 0, the PadFlag Status is cleared and the corresponding bit in a variable called PadError Status is set. This means that there is an error in detection as the press of the pad is too long.

Because all this happens over a series of Time Slices, the application needs to check that PadFlags is 0 before any PadError or PadResult is valid. A non-zero PadFlag variable indicates that pads are still being processed.

PadFlag(F)

Pad Flag Status (IRAM address 0x2F)

Reset Value: 00_H

7	6	5	4	3	2	1	0
Pad Line7	Pad Line6	Pad Line5	Pad Line4	Pad Line3	Pad Line2	Pad Line1	Pad Line0
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pad Line0	0	rw	Pad Line 0 of PadFlag 0B , Pad not touched 1B , Pad touched
Pad Line1	1	rw	Pad Line 1 of PadFlag 0B , Pad not touched 1B , Pad touched
Pad Line2	2	rw	Pad Line 2 of PadFlag 0B , Pad not touched 1B , Pad touched

ROM Library

Field	Bits	Type	Description
Pad Line3	3	rw	Pad Line 3 of PadFlag 0B , Pad not touched 1B , Pad touched
Pad Line4	4	rw	Pad Line 4 of PadFlag 0B , Pad not touched 1B , Pad touched
Pad Line5	5	rw	Pad Line 5 of PadFlag 0B , Pad not touched 1B , Pad touched
Pad Line6	6	rw	Pad Line 6 of PadFlag 0B , Pad not touched 1B , Pad touched
Pad Line7	7	rw	Pad Line 7 of PadFlag 0B , Pad not touched 1B , Pad touched

PadResult(R)
PadResult Status (IRAM address 0x2E)
Reset Value: 00_H

7	6	5	4	3	2	1	0
Pad Line7	Pad Line6	Pad Line5	Pad Line4	Pad Line3	Pad Line2	Pad Line1	Pad Line0
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pad Line0	0	rw	Pad Line 0 of PadResult 0B , Pad not valid 1B , Pad valid
Pad Line1	1	rw	Pad Line 1 of PadResult 0B , Pad not valid 1B , Pad valid
Pad Line2	2	rw	Pad Line 2 of PadResult 0B , Pad not valid 1B , Pad valid

ROM Library

Field	Bits	Type	Description
Pad Line3	3	rw	Pad Line 3 of PadResult 0B , Pad not valid 1B , Pad valid
Pad Line4	4	rw	Pad Line 4 of PadResult 0B , Pad not valid 1B , Pad valid
Pad Line5	5	rw	Pad Line 5 of PadResult 0B , Pad not valid 1B , Pad valid
Pad Line6	6	rw	Pad Line 6 of PadResult 0B , Pad not valid 1B , Pad valid
Pad Line7	7	rw	Pad Line7 of PadResult 0B , Pad not valid 1B , Pad valid

PadError(E)
PadError Status (IRAM address 0x2D)
Reset Value: 00_H

7	6	5	4	3	2	1	0
Pad Line7	Pad Line6	Pad Line5	Pad Line4	Pad Line3	Pad Line2	Pad Line1	Pad Line0
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pad Line0	0	rw	Pad Line 0 of PadError 0B , Pad OK 1B , Pad ERROR
Pad Line1	1	rw	Pad Line 1 of PadError 0B , Pad OK 1B , Pad ERROR
Pad Line2	2	rw	Pad Line 2 of PadError 0B , Pad OK 1B , Pad ERROR

Field	Bits	Type	Description
Pad Line3	3	rw	Pad Line 3 of PadError 0B , Pad OK 1B , Pad ERROR
Pad Line4	4	rw	Pad Line 4 of PadError 0B , Pad OK 1B , Pad ERROR
Pad Line5	5	rw	Pad Line 5 of PadError 0B , Pad OK 1B , Pad ERROR
Pad Line6	6	rw	Pad Line 6 of PadError 0B , Pad OK 1B , Pad ERROR
Pad Line7	7	rw	Pad Line 7 of PadError 0B , Pad OK 1B , Pad ERROR

23.2.2.2 Implementation Details of Function

The important formulas used in the function are how the Total_TSCTRL/H obtained, how Average is calculated, how LowTrip (Trip point) is derived and finally how a comparison is made for a touch or no-touch of a pad. SFR LTS_TSCTR value is used for this calculation.

Total number of samples accumulated is defined in AccumulatorCounter input. Total value of all samples is TOTAL_TSCTRL/H. n is user-defined input for FINDTOUCHEDPAD function - AccumulatorCounter.

$$\text{TOTAL_TSCTRL/H}(x) = \sum_{i=1}^{\text{AccumulatorCounter}} \text{LTS_TSCTR}(i) \quad (23.7)$$

where i is the user-defined input AccumulatorCounter (number of samples required), supporting value 1 to 255.

The Average is calculated as follows:

$$\text{AVERAGEL/H}(x) = \text{AVERAGEL/H}(x-1) + \text{TOTAL_TSCTRL/H}(x) - \frac{\text{AVERAGEL/H}(x-1)}{2^n} \quad (23.8)$$

where n is the user-defined input Divisor n , supporting value 1 to 8.

The LowTrip (Trip point) value is calculated by subtracting user-defined input subtraction m from the Average.

(23.9)

$$\text{LOWTRIPL}/H(x) = \text{AVERAGEL}/H(x) - \text{SUBTRACTION } m$$

With the LowTrip value calculated from the Average value, the comparison is done with the current accumulated LTS_TSCTR values in TOTAL_TSCTRL/H * 2ⁿ where n is the user-defined input Divisor n.

A padtouch is identified when:

(23.10)

$$\text{TOTAL_TSCTRL}/H(x) \times 2^n < \text{LOWTRIPL}/H(x)$$

A pad not touched or a padtouch being removed/lifted are identified when:

(23.11)

$$\text{TOTAL_TSCTRL}/H(x) \times 2^n \geq \text{LOWTRIPL}/H(x)$$

When padtouch is identified, a counter (PDC) is initialized to 0xFF and will start decrementing till the padtouch is being removed/lifted. At this instant, the PDC is compared with ShortCount (input) to determine if the padtouch is a valid or invalid touch.

Internal states

Because of the time multiplexed nature of the touch-sense functionality, the function can be in a number of states when entered and the states can be modified inside this function and outside by the user application. Here is a state diagram that shows the various states and how to transit from one state to the other. There are 5 states in total: Idle State, Pad Touched State, Pad Released State, Error State and Result State.

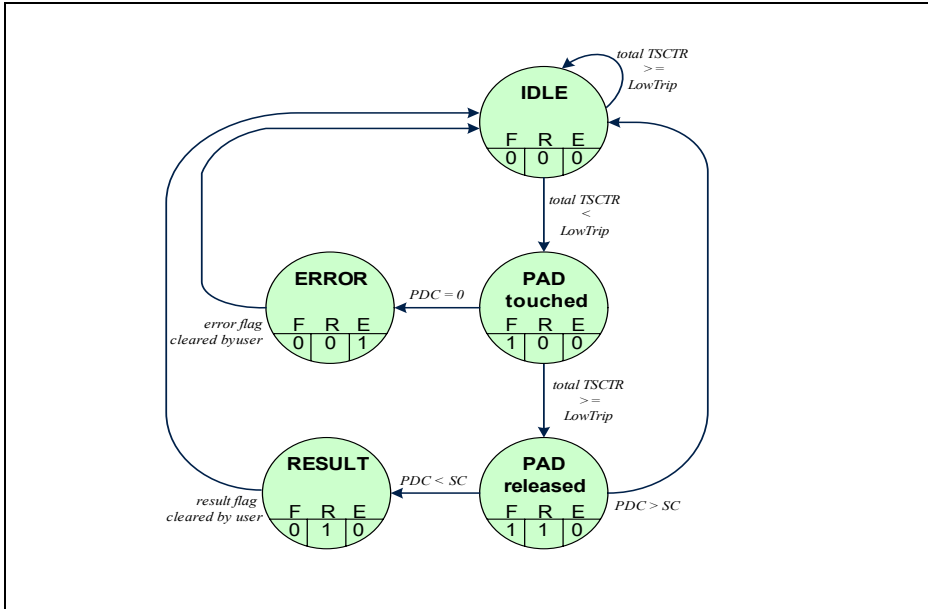


Figure 23-8 State diagram

Actions on entering a state:

IDLE:

On Enter: check if $\text{Total_TSCTRL/H} < \text{LowTripL/H}$ point for this pad turn

On Exit: If yes, set the corresponding pad flag, start the counter ($\text{PDC} = 0\text{xFF}$)

Pad Touched:

On Enter: decrement the PDC, check if $\text{PDC} == 0$, check if $\text{Total_TSCTRL/H} < \text{LowTripL/H}$ point for this pad turn

On Exit: depending on the results of the checks, set relevant pad error flag, set relevant pad result flag

Pad Released:

On Enter: check if $\text{PDC} \geq \text{short count}$, check $\text{PDC} < \text{short count}$

On Exit: clear all flags if $\geq \text{short count}$, clear pad flag if $< \text{short count}$

Pad Result:

On Enter: check if pad result is cleared by the user, if not, exit.

On Exit: IDLE

Pad Error:

On Enter: check if pad error is cleared by the user, if not, exit.

On Exit: IDLE

Pad Down Counter - PDC

When function detects an initial padtouch, it will initialize the PDC value to 0xFF. Every time function is called, as long as pad is still being touched, function will decrement PDC value. In case of dual pads, PDC will only be decremented once during both pads' analysis time (able to detect that PDC is already decremented before). The diagram below shows how the Pad Down Counter PDC works, this is a count-down software counter. Time Slice/Frame interrupt is its clock.

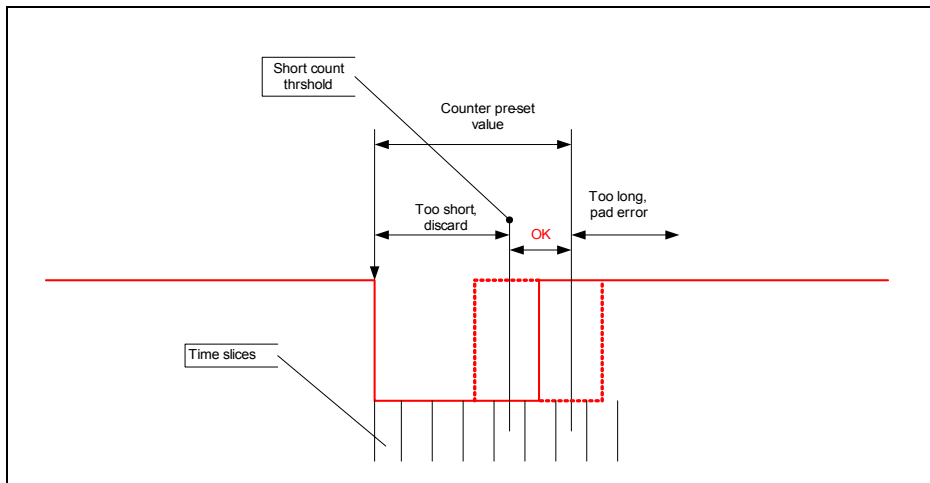


Figure 23-9 Pad Down Counter - PDC

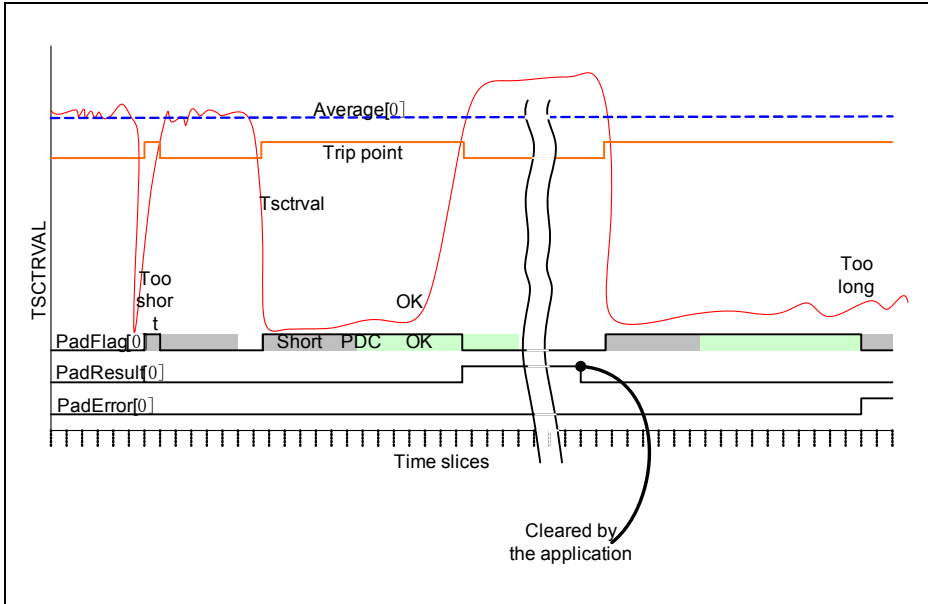


Figure 23-10 Timing Diagram

Padline 0 used for the example above to show the interaction between PadFlags, PadResult, PadError and PDC. The trip point (LowTrip) can be fixed or as in this illustration, hysteresis can be added whenever a padflag is set (by changing the user-input subtraction m)

The PDC gets pre-loaded with its time-out value on the first PadFlag being set. In any subsequent PadTurn this counter decrements until either it reached 0 or until all PadFlags are cleared where it is then set to 0. If during a PadTurn the last remaining set PadFlag gets cleared the PDC gets set to 0 on exit of the function. This can happen either when a pad is touched too briefly (short count) or when the last pad is released before PDC times out.

A corner case can arise when the user does not clear the error or result before another pad is touched. In such a situation it is possible that more than two bits in the result can show up if for example the first valid result was a combination pad being touched and the next one touched was a single pad whose padline was not part of the previous combination pad.

PDC is decremented once every 1 period (All Enabled Pads Accumulated Count Period), even if dual padflags maybe set.

Time Slice, Time Frame and Period Definition

There is a Time Slice, Time Frame and Periods naming convention. There is hardware-controlled scheme period and software-controlled ROM Library scheme period (Single Pad Accumulated Count Period and All Enabled Pads Accumulated Count Period). The timing definitions, when Touch-sense is enabled, are shown in the equations below while an example of their relationships are portrayed in [Figure 23-11](#) for a clearer picture.

A Time Slice is calculated as follows:

$$\text{Time Slice} = \frac{\text{Prescaler} \times 256}{f_{\text{CLK}}} \quad (23.12)$$

where prescaler is the LEDTS-Counter Clock Pre-Scale Factor (LTS_GLOBCL0.CLK_PS) and the input clock f_{CLK} will be either 8 MHz or 24 MHz, depending on the USER_ID setting.

A Time Frame is defined as follows:

$$\text{Time Frame} = \text{Time Slice} \times (\text{No. of LED enabled} + 1) \quad (23.13)$$

A Period (Hardware-controlled scheme) is defined as follows:

$$\text{Period}_{(\text{Hardware-Controlled Scheme})} = \text{Time Frame} \times \text{No. of PADTx enabled} \quad (23.14)$$

For software-controlled ROM Library Scheme, the Single Pad Accumulated Count Period is defined as follows:

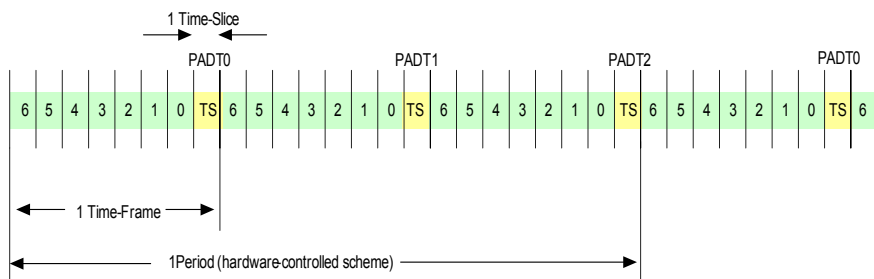
$$\text{Period}_{(\text{Single Pad Accumulated Count})} = \text{Time Frame} \times (\text{AccumulatorCount} + 1) \quad (23.15)$$

where AccumulatorCount is the user-defined input for FINDTOUCHEDPAD function.

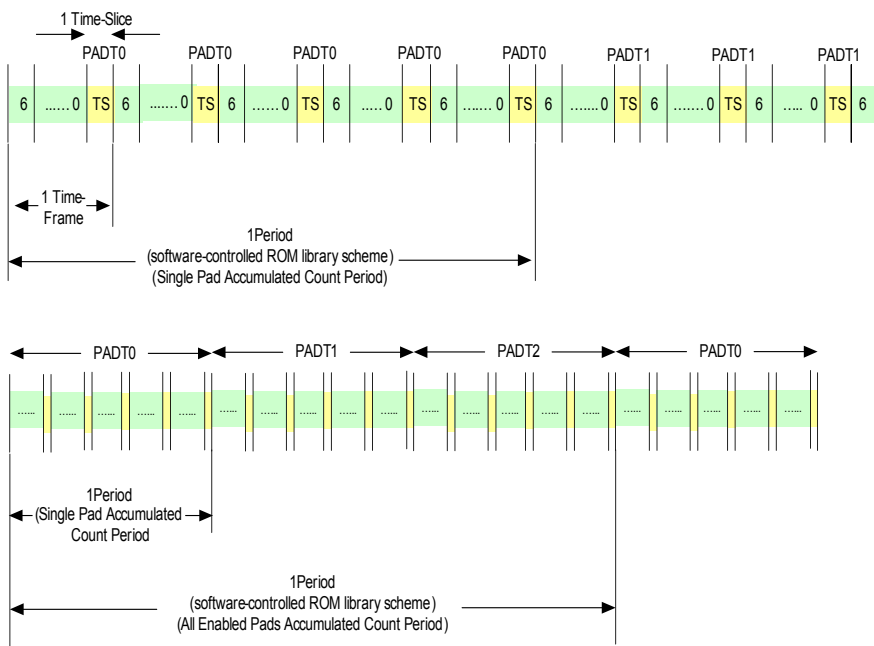
For software-controlled ROM Library Scheme, the All Enabled Pads Accumulated Count Period is defined as follows:

$$\text{Period}_{(\text{All Enabled Pads})} = \text{Period}_{(\text{Single Pad Accumulated Count})} \times \text{No. of PADTx enabled} \quad (23.16)$$

For example, 7 LEDs and 3 Touch Sense Pad Turns are enabled



For example, 7 LEDs and 3 Touch Sense Pad Turns are enabled, AccumulatorCounter = 0x04



PDC decrements once each time in 1 period (All Enabled PADTx Accumulated Count Period) where a pad is touched, in the case of a combination pad (2 pads touched), the PDC decrements once also in 1 Period (All Enabled PADTx Accumulated Count Period)

Figure 23-11 Period Relationship for Software-Controlled ROM Library Scheme

23.2.3 Use of the functions in Interrupts

With correct inputs prepared for the functions, the functions can be called from Time Slice interrupt or Time Frame interrupt, depending on whether LED or/and TS is enabled. **Figure 23-12** and **Figure 23-13** illustrate how users can use and call the functions in respective situations.

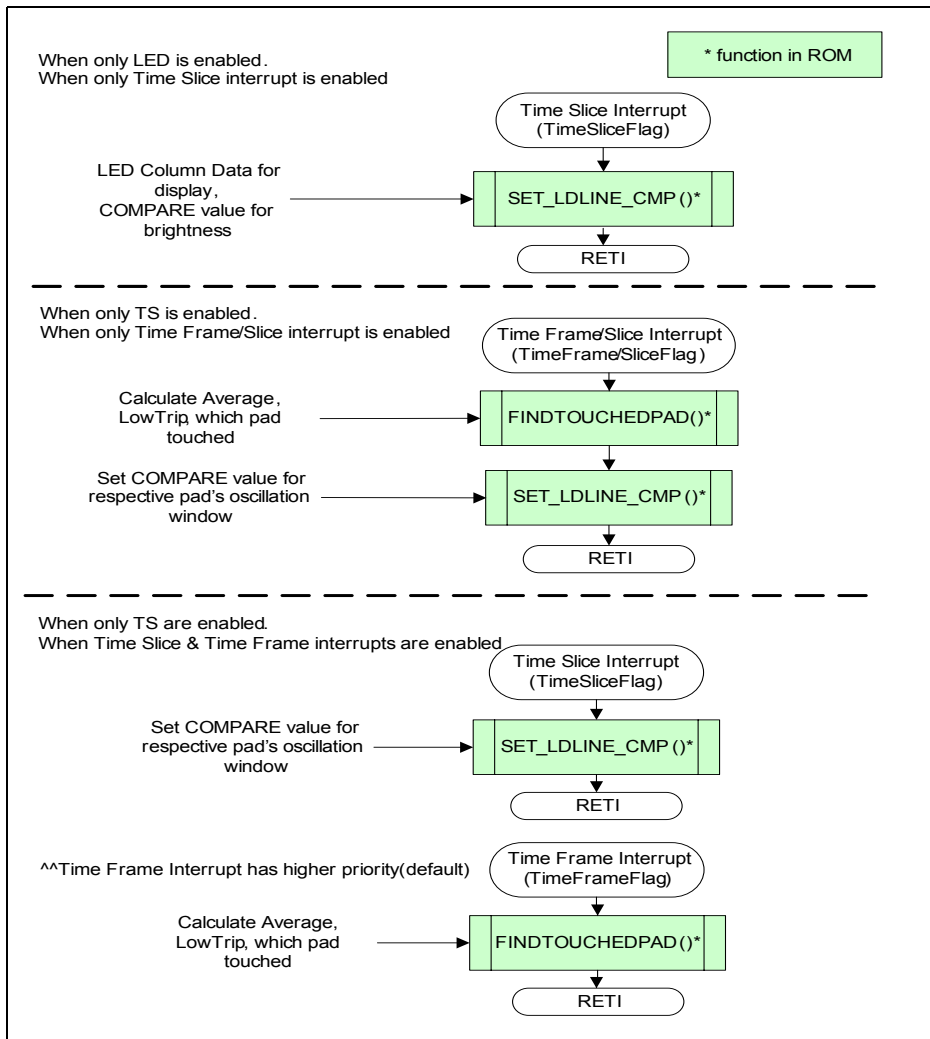


Figure 23-12 Calling of Functions in Interrupt Routine (i)

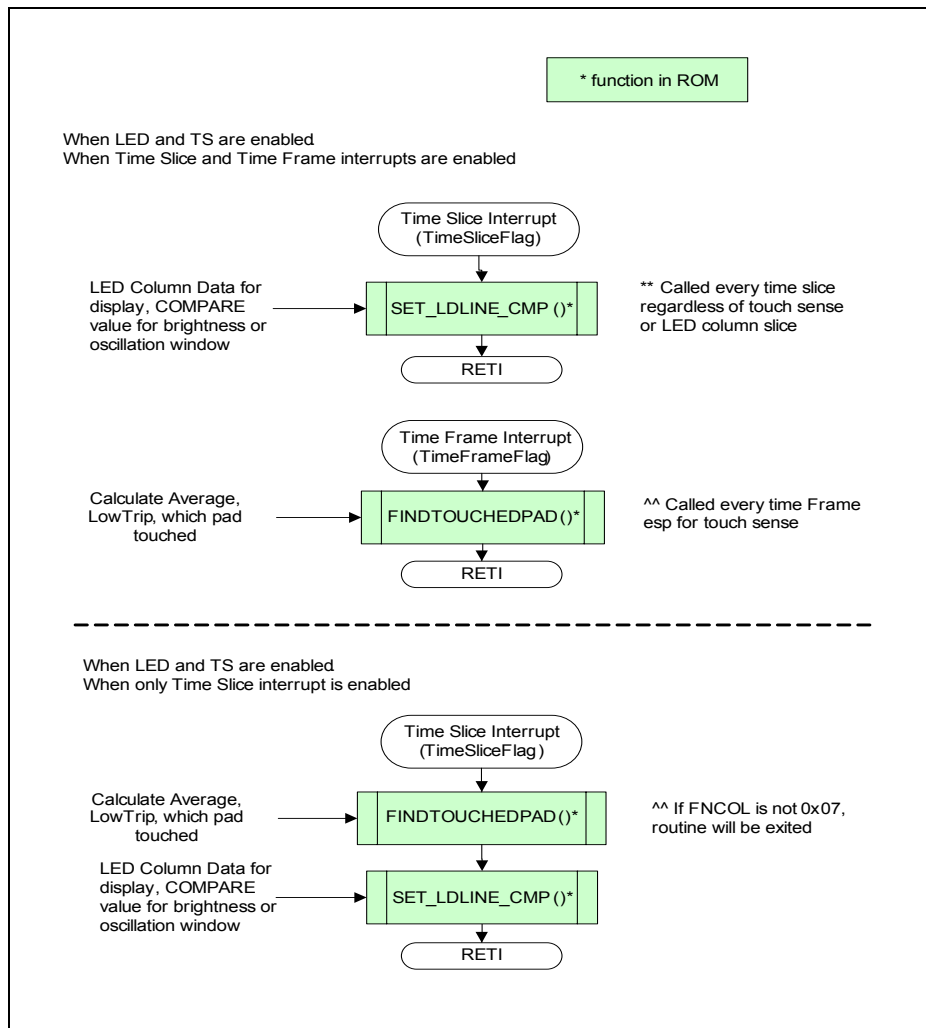


Figure 23-13 Calling of Functions in Interrupt Routine (ii)

23.3 MDU ROM Library (MATH Function)

The MDU ROM Library (MATH Function) contains 4 routines that reside in ROM that are accessible to users. The call functions for MDU ROM Library is listed in [Table 23-14](#)

Table 23-14 MDU ROM Library (MATH Function) Routine Table

Address	Name	Description
0xDFC0	?C?IMUL_XC	16-bit Multiplication
0xDFC3	?C?LMUL_XC	32-bit Multiplication
0xDFC6	?C?UIDIV_XC	16/16-bit Division
0xDFC9	?C?ULDIV_XC	32/32-bit Division

23.3.1 Integer Multiplication

(23.17)

Output is Result of $\text{Value1} \times \text{Value2}$

Table 23-15 Specifications of 16-bit Multiplication Subroutine

Subroutine	?C?IMUL_XC
Address	DFC0 _H
Input	R7 of current Register Bank: Value1 (Low Byte)
	R6 of current Register Bank: Value1 (High Byte)
	R5 of current Register Bank: Value2 (Low Byte)
	R4 of current Register Bank: Value2 (High Byte)
Output	R7 of current Register Bank: Result of $[\text{Value1} \times \text{Value2}]$ (Low Byte)
	R6 of current Register Bank: Result of $[\text{Value1} \times \text{Value2}]$ (High Byte)
Stack size required	2
Resource used/destroyed	A, MD0, MD1, MD4, MD5, MR0, MR1, MR2, MR3, MDUCON

23.3.2 Long Multiplication

(23.18)

Output is Result of $\text{Value1} \times \text{Value2}$

Table 23-16 Specifications of 32-bit Multiplication Subroutine

Subroutine	?C?LMUL_XC
Address	DFC3 _H
Input	R7 of current Register Bank: Value1 (Bits 7-0)
	R6 of current Register Bank: Value1 (Bits 15-8)
	R5 of current Register Bank: Value1 (Bits 23-16)
	R4 of current Register Bank: Value1 (Bits 31-24)
	R3 of current Register Bank: Value2 (Bits 7-0)
	R2 of current Register Bank: Value2 (Bits 15-8)
	R1 of current Register Bank: Value2 (Bits 23-16)
	R0 of current Register Bank: Value2 (Bits 31-24)
Output	R7 of current Register Bank: Result of $[\text{Value1} \times \text{Value2}]$ (Bits 7-0)
	R6 of current Register Bank: Result of $[\text{Value1} \times \text{Value2}]$ (Bits 15-8)
	R5 of current Register Bank: Result of $[\text{Value1} \times \text{Value2}]$ (Bits 23-16)
	R4 of current Register Bank: Result of $[\text{Value1} \times \text{Value2}]$ (Bits 31-24)
Stack size required	2
Resource used/destroyed	A, MD0, MD1, MD4, MD5, MR0, MR1, MR2, MR3, MDUCON, MDUSTAT

23.3.3 Integer Division

(23.19)

Outputs are Result and Remainder of $\frac{\text{Value1}}{\text{Value2}}$

Table 23-17 Specifications of 16/16-bit Division Subroutine

Subroutine	?C?UIDIV_XC
Address	DFC6 _H
Input	R7 of current Register Bank: Value1 (Low Byte)
	R6 of current Register Bank: Value1 (High Byte)
	R5 of current Register Bank: Value2 (Low Byte)
	R4 of current Register Bank: Value2 (High Byte)
Output	R7 of current Register Bank: Result of [Value1 ÷ Value2] (Low Byte)
	R6 of current Register Bank: Result of [Value1 ÷ Value2] (High Byte)
	R5 of current Register Bank: Remainder of [Value1 ÷ Value2] (Low Byte)
	R4 of current Register Bank: Remainder of [Value1 ÷ Value2] (High Byte)
Stack size required	2
Resource used/destroyed	A, MD0, MD1, MD4, MD5, MR0, MR1, MR4, MR5, MDUCON, MDUSTAT

23.3.4 Long Division

(23.20)

Outputs are Result and Remainder of $\frac{\text{Value1}}{\text{Value2}}$

Table 23-18 Specifications of 32/32-bit Division Subroutine

Subroutine	?C?ULDIV_XC
Address	DFC9 _H
Input	R7 of current Register Bank: Value1 (Bits 7-0)
	R6 of current Register Bank: Value1 (Bits 15-8)
	R5 of current Register Bank: Value1 (Bits 23-16)
	R4 of current Register Bank: Value1 (Bits 31-24)
	R3 of current Register Bank: Value2 (Bits 7-0)
	R2 of current Register Bank: Value2 (Bits 15-8)
	R1 of current Register Bank: Value2 (Bits 23-16)
	R0 of current Register Bank: Value2 (Bits 31-24)

Table 23-18 Specifications of 32/32-bit Division Subroutine (cont'd)

Output	R7 of current Register Bank: Result of $[\text{Value1} \div \text{Value2}]$ (Bits 7-0)
	R6 of current Register Bank: Result of $[\text{Value1} \div \text{Value2}]$ (Bits 15-8)
	R5 of current Register Bank: Result of $[\text{Value1} \div \text{Value2}]$ (Bits 23-16)
	R4 of current Register Bank: Result of $[\text{Value1} \div \text{Value2}]$ (Bits 31-24)
	R3 of current Register Bank: Remainder of $[\text{Value1} \div \text{Value2}]$ (Bits 7-0)
	R2 of current Register Bank: Remainder of $[\text{Value1} \div \text{Value2}]$ (Bits 15-8)
	R1 of current Register Bank: Remainder of $[\text{Value1} \div \text{Value2}]$ (Bits 23-16)
	R0 of current Register Bank: Remainder of $[\text{Value1} \div \text{Value2}]$ (Bits 31-24)
Stack size required	2
Resource used/destroyed	A, MD0, MD1, MD2, MD3, MD4, MD5, MR0, MR1, MR2, MR3, MR4, MR5, MDUCON, MDUSTAT, DPL, DPH
	R0 - R7 of current Register Bank

23.4 EEPROM Emulation ROM Library

The XC82x provide EEPROM emulation functionality via the ROM library. The EEPROM is emulated using the on-chip flash memory. The ROM library provide a framework to access the emulated EEPROM.

The ROM library used the following application note as a reference, AP0805710 XC866/XC886/XC888 EEPROM EMULATION.

23.4.1 Feature

The EEPROM emulation ROM library has the following feature

- Provide functions to initialize, fix, read and write to emulated EEPROM
- Support EEPROM emulation size of 31, 62, 93 or 124 bytes
- Up to 1.6 millions endurance cycles for a data retention time of 2 years

23.4.2 System requirement

The ROM library has several requirement

- Use 512 bytes of flash memory from address AE00_H to AFFF_H for EEPROM emulation¹⁾
- Only support KEIL C51 toolchain (small memory model, big endian, register calling convention)
- Only support polling based flash operation
- No support for detection of aborted programming operation

23.4.3 Concept

EEPROM emulation is implemented using 512 bytes of flash that is divided into 2 logical sectors. At any one time only one of the logical sectors is the “active sectors” i.e. the sector containing the latest dataset. User application needs to initialise the emulated EEPROM before it can be used. This can be done by calling the initialisation function and specifying the desired emulation scheme. The initialisation function will update the information in **“EEPROMInfo data structure” on Page 23-46** that is used by the ROM library functions.

The ROM library supports 4 emulation scheme i.e. Mode 0, Mode 1, Mode 2 & Mode 3. Basically each mode represent the size of the emulated EEPROM.

User application will access the emulated EEPROM using logical address. The number of valid logical address is dependent on the size of the emulated EEPROM. Each logical address represent 32 bytes of data where 31 bytes are user data and one status byte. This represent the required size for each flash write operation i.e. flash word line (WL) size.

1) For 4K device, EEPROM is emulated using address 0E00_H to 0FFF_H

ROM Library

The logical address provides a convenient abstraction so that user application don't need to deal with the actual physical flash address that is used for emulation.

The status byte is used by the ROM library to manage the EEPROM emulation. It is written at the end of each flash wordline. The user application can safely ignore this status byte.

Table 23-19 EEPROM emulation scheme

Emulation Scheme	Emulated EEPROM size (byte)	Valid logical address
Mode_0	31 data + 1 status	WL_0
Mode_1	62 data + 2 status	WL_0 & WL_1
Mode_2	93 data + 3 status	WL_0, WL_1 & WL_2
Mode_3	124 data + 4 status	WL_0, WL_1, WL_2 & WL_3

Table 23-20 Relationship between logical address and data bytes

Logical address	Data bytes
WL_0	0 to 30
WL_1	31 to 61
WL_2	62 to 92
WL_3	93 to 124

Additionally the initialisation function can perform some basic error detection. It can detect if an abort occur during a flash erase operation or emulated EEPROM is used for first time. The initialisation function assumes that flash area used for EEPROM emulation is already erased when its used for the first time.

User application will access the emulated EEPROM using logical address via the read and write function. Each read operation will read out 31 user data bytes and 1 status byte. The 32 bytes of data will be stored in IRAM location provided by the calling function.

Similarly each write operation can only program 31 bytes and 1 status byte. The data to be written should be stored in IRAM location provided by the calling function. The status byte value would be automatically be updated by the write function.

Once the "active sector" is completely filled with data, subsequent write operation will be in the next logical sector. The next sector is set as the "active sector" and the previous "active sector" is reclaimed. The reclaim operation involves, programming all valid data from the previous "active sector" to the current "active sector" before erasing the previous "active sector".

To update less than 31 data bytes in the emulated EEPROM, user application needs to perform a read to the logical address where the data resides. Once the read data are

stored in the IRAM, user application will then need to update the relevant data in IRAM with the desired data. This is then followed by a call to write function to program the data into the emulated EEPROM.

23.4.4 API description

The EEPROM emulation operation is controlled via one data structure and 4 functions. Additionally some definition is provided for ease of use.

23.4.4.1 Constant definition

Several constant definition was added to give meaningful name and promote code readability

EEPROM Emulation scheme

- #define MODE_0 32
- #define MODE_1 64
- #define MODE_2 96
- #define MODE_3 128

Logical Address

- #define WL_0 0
- #define WL_1 1
- #define WL_2 2
- #define WL_3 3

23.4.4.2 EEPROMInfo data structure

The EEPROM emulation API use a data structure called *EEPROMInfo* to manage the EEPROM emulation operation.

```
typedef struct EEPROMInfo{
    unsigned int ActiveSector;
    unsigned int WriteAddress;
    unsigned char DataSize;
}data EEPROMInfo;
```

EEPROMInfo is a structure containing 3 variables, *ActiveSector*, *WriteAddress* and *DataSize*. The structure must be declared by user application and must reside in indirect IRAM address.

ActiveSector store the current "active sector" address. It is used by *ReadEEPROM()* to find the current emulated dataset. *WriteAddress* store the start address to write new data set when user application calls *WriteEEPROM()*. Both variable is updated by call to

InitEEPROM() and *WriteEEPROM()*. *DataSize* store the EEPROM emulation dataset size

EEPROMInfo should not be modified directly by user. It only should be modified by functions provided by EEPROM emulation API.

23.4.4.3 Functions

The EEPROM emulation API provide 4 functions

- InitEEPROM
- FixEEPROM
- WriteEEPROM
- ReadEEPROM

These functions need to be used when accessing the emulated EEPROM.

InitEEPROM

Table 23-21 InitEEPROM

Function name	InitEEPROM
Function prototype	unsigned char InitEEPROM(unsigned char mode, EEPROMInfo *config)
Input	<p>mode - Emulation scheme i.e. size of emulated EEPROM. Valid value is 32, 64, 96 and 128</p> <p>config - Pointer to <i>EEPROMInfo</i> data structure</p>
Return	<p>status - Status of emulated EEPROM. Input to FixEEPROM()</p> <p>0x0 - Status of emulated EEPROM is OK</p> <p>0x1 - Sector 8 & 9 needs to be erased</p> <p>0x2 - Sector 6 & 7 needs to be erased</p> <p>0x3 - Sector 6 to 9 needs to be erased</p>
Max stack size	4 bytes

InitEEPROM() will initialize the EEPROMInfo data structure based on the selected emulation mode. It should be called at least once before using the other API functions. InitEEPROM() assume that flash area used for EEPROM emulation is in erased condition when InitEEPROM() is called for the first time.

It will update the EEPROM configuration to indicate the current "active sector" and the flash address that is to be written to on subsequent call to WriteEEPROM().

ROM Library

The “active sector” is selected based on the sector containing a valid status byte. If both logical sector contains valid status byte, the one containing the least amount of valid status byte is selected as the active sector. Additionally InitEEPROM() will return an error status requesting the sector with the most valid status byte to be erased. However there's a scenario where this might not be desired e.g. if there was a sudden power off during a WriteEEPROM() reclaim operation.

For example a 124 bytes configured EEPROM is powered off suddenly during the WriteEEPROM() reclaim operation. Thus not all the 124 bytes of valid information is copied to the new sector.

Once the “active sector” has been determined, the flash address to be written is determined as the first flash WL without a valid status byte in the “active sector”.

InitEEPROM() will also check for situation where erase operation has been unexpectedly aborted and return a status byte indicating sectors that needs to be re-initialised.

FixEEPROM
Table 23-22 FixEEPROM

Function name	FixEEPROM
Function prototype	<code>void FixEEPROM(unsigned char sector_status, unsigned char idata *buffer)</code>
Input	sector_status - status returned by InitEEPROM()
	buffer - Pointer to 32 bytes buffer in IRAM location
Return	None
Max stack size	4 bytes + 12 bytes (Boot ROM User Routines)

FixEEPROM() will take the status byte returned by InitEEPROM() and fix the invalid sectors. It will erase the flash sectors and write a status byte to indicate that the sector has been erased.

A pointer to a buffer of 32 bytes in IRAM is needed to enable flash write operation. For save operation ensure that flags in NMISR SFR are cleared before calling this function.

WriteEEPROM

Table 23-23 WriteEEPROM

Function name	WriteEEPROM
Function prototype	void WriteEEPROM(unsigned char address, char idata * src, EEPROMInfo *config)
Input	address - Logical address of emulated EEPROM to write to
	src - Pointer to 32 bytes of data in IRAM that is to be written
	config - Pointer to <i>EEPROMInfo</i> data structures
Return	None
Max. stack size	6 bytes + 12 bytes (Boot ROM User Routines)

WriteEEPROM() will perform the write operation based on EEPROM configuration. It will program to the valid flash address and update the EEPROM configuration. Each write operation is 32 bytes (31 data bytes + 1 status byte).

Additionally it will "reclaim" the previous "active sector" when its filled with data. For save operation ensure that all NMISR SFR flags are cleared before calling this function.

ReadEEPROM

Table 23-24 ReadEEPROM

Function name	ReadEEPROM
Function prototype	unsigned char ReadEEPROM(unsigned char address, char idata * dst, EEPROMInfo *config)
Input	address - Logical address of emulated EEPROM to read from
	dst - Pointer to 32 bytes IRAM buffer to store read data
	config - Pointer to <i>EEPROMInfo</i> data structure
Output	status - Read operation result
	0x00 - Read operation successful
	0xFF - Read operation fail
Max stack size	2 bytes

ReadEEPROM() will search the current "active sector" for the most current data based on the given "address". It will always return 32 bytes of data i.e. 31 data byte + 1 status bytes.

The "address" takes the form of a number from 0 to 3. For dataset size of 32bytes, only address 0 is valid. For dataset size of 128, address 0 to 3 is valid. Address 2, will return byte 64 to byte 95 where byte 95 is the status bytes.

Function will return an error code if the address can't be found.

23.4.4.4 Example of API usage

Here's an example in 'C' that use the EEPROM emulation API. The example shows how to initialise the flash area for use and some error handling mechanism.

```
#include "EEPROM.h"
```

```
void main(void)
{
    /*EEPROM emulation required data structure*/
    char idata buffer[32];
    EEPROMInfo config;
    unsigned char eeprom_status;

    /*Initialised and check emulated EEPROM integrity*/
    eeprom_status = InitEEPROM(MODE_3, &config);
    if(0 != eeprom_status){
        if( 0x0 == config.ActiveSector){
            /*Erase abort detected or first time using EEPROM*/
            /*Initialise invalid sectors*/
            FixEEPROM(eeprom_status, buffer);
        }else{
            /*Programming or erasing aborted suddenly during reclaiming
            operation */

            /*Depending on user application, additional action can be taken*/

            /*Initialise invalid sectors. Data would be erased*/
            FixEEPROM(eeprom_status, buffer);
        }
    }
}
```

```
}

/*Check that EEPROM sectors are in valid condition*/
eeprom_status = InitEEPROM(MODE_3, &config);
if (0 != eeprom_status){
    /*Fatal failure in EEPROM*/
    while(1);
}

}

/*User application code*/

/*Example of reading and writing to emulated EEPROM*/
/*Get current data*/
ReadEEPROM(WL_1, &buffer, &config);
/*Update data to store and write to EEPROM*/
buffer[20] += 1;
/*Write back updated data*/
WriteEEPROM(WL_1, &buffer, &config);
while(1); /*Wait for power off*/
}
```

Keyword Index

A

ADC

- Accumulated Application Read View 21-65
- Alias Feature 21-54
- Boundary Flag 21-53
- Channel Scan Request Source Handling 21-19
- Channel Scan Source 21-18
- Clocking Scheme 21-13
- Conversion Timing 21-61
- Data Reduction and Filtering 21-80
 - Digital Low Pass Filter Mode 21-82
 - Standard Data Reduction Mode 21-81
- Differential Like Measurement 21-46
- Digital Low Pass Filtered Application Read View 21-66
- Error Definitions 21-12
 - Differential non linearity error (DNL) 21-12
 - Gain Error 21-12
 - Integral non linearity error (INL) 21-12
 - Offset Error 21-12
 - Total unadjusted error (TUE) 21-12
- Interrupt Request Handling 21-83
 - Channel Events 21-83
 - Out of Range Comparator Events 21-83
 - Request Source Events 21-83
 - Result Events 21-83
- Limit Checking 21-52
- Out of Range Comparator (ORC) 21-57
- Out of range comparator (ORC) 21-9
- Queued Request Source Handling 21-25
- Queued Source 21-18
- Reference Selection 21-45

Registers

- CHINCR 21-89
- CHINFR 21-87
- CHINSR 21-88
- CRCR1 21-23
- CRMR1 21-21
- CRPR1 21-24
- ENORC 21-58
- ETRCR 21-37
- EVINCR 21-86
- EVINFR 21-84
- EVINSR 21-85
- GLOBCTR 21-14
- GLOBSTR 21-15
- INPCR0 21-51
- LCBR0 21-54
- LCBR1 21-54
- LORC 21-60
- PRAR 21-41
- Q0R0 21-34
- QBUR0 21-36
- QINR0 21-33
- QMR0 21-29
- QSR0 21-31
- RCRx (x = 0 - 3) 21-77
- RESR_xH (x = 0 - 3) 21-72
- RESR_xL (x = 0 - 3) 21-69, 21-72, 21-75
- VFCR 21-79
- Request Source Arbitration 21-38
- Standard Application Read View 21-65

B

- Bitaddressable 3-10
- BMI 5-1
- Boot and Startup 5-1
- Boot ROM 3-1
- Boot ROM operating mode 5-4
 - Boot-Loader Mode 5-4
 - OCDs mode 5-5
 - User mode (diagnostic) 5-4

User mode (productive) 5-4
 Boot-loader 4-6, 4-11, 5-4
 Brownout reset 7-7
 Buffer mechanism 4-3

C

CCU6

Block Diagram 20-3
 Hall Sensor Mode 20-90
 Interrupt Handling 20-110
 Multi-Channel Mode 20-87

Registers

CC63RH **20-83**
 CC63RL **20-83**
 CC63SRH **20-84**
 CC63SRL **20-84**
 CC6xRH **20-46**
 CC6xRL **20-46**
 CC6xSRH **20-47**
 CC6xSRL **20-47**
 CMPMODIFH **20-54**
 CMPMODIFL **20-53**
 CMPSTATL **20-51, 20-52**
 IENL **20-120, 20-121**
 INPH **20-125**
 INPL **20-124**
 ISL **20-112, 20-113**
 ISRL **20-118**
 ISSL **20-116**
 MCMCTR **20-104**
 MCMOUTH **20-109**
 MCMOUTL **20-107**
 MCMOUTSH **20-106**
 MCMOUTSL **20-106**
 MODCTRH **20-99**
 MODCTRL **20-98**
 Offset addresses 20-5
 Overview **20-4**
 PISEL0H **20-128**
 PISEL0L **20-127**
 PISEL2 **20-129**
 PSLR **20-103**
 T12DTCH **20-49**

T12DTCL **20-49**
 T12H **20-43**
 T12L **20-43**
 T12MSELH **20-55**
 T12MSELL **20-55**
 T12PRH **20-44**
 T12PRL **20-44**
 T13H **20-79**
 T13L **20-79**
 T13PRH **20-81**
 T13PRL **20-81**
 TCTR0H **20-59**
 TCTR0L **20-57**
 TCTR2H **20-63**
 TCTR2L **20-61**
 TCTR4H **20-65**
 TCTR4L **20-64**
 TRPCTRH **20-101**
 TRPCTRL **20-100**

T12 20-17

Capture Modes 20-37
 Compare Mode 20-25
 Counting Scheme 20-21
 Dead-Time Generation 20-32
 Hysteresis-like Control Mode 20-31
 Output Modulation and Level Selection 20-35
 Shadow Register Transfer 20-41

T13 20-67

Compare Mode 20-74
 Counting Scheme 20-70
 Shadow Register Transfer 20-77
 Synchronization to T12 20-72

Trap Handling 20-85

Chip identification number 1-12
 Circular stack memory 4-3
 Correction algorithm 4-9
 Count Clock 14-9
 Counter 13-1
 CPU 2-1

Functional Blocks 2-1
 Instruction Table 2-10
 Instruction Timing 2-8

Registers 2-3

- ACC 2-3
- B Register 2-3
- Data Pointer 2-3
- PCON 2-6
- PSW 2-3
- Stack Pointer 2-3

D

- Data memory 3-2
- Debug System 10-1
 - Functional Overview 10-5
 - Breakpoints 10-5
 - Debug Suspend Control 10-7
 - Monitor Program 10-6
 - Monitor Running 10-7
 - Return to User Program 10-7
 - Single Step 10-7

Document

- Acronyms 1-14
- Terminology 1-14
- Textual conventions 1-13

Dynamic error detection 4-9
E

- EEPROM emulation 4-3
- Embedded voltage regulator 7-1
 - Low power voltage regulator 7-2
 - Main voltage regulator 7-2
 - Threshold voltage levels 7-2
- Error Correction Code 4-9
- External data memory 3-2

F

- Flash 4-1
 - Endurance 4-4
 - Erase mode 4-8
 - Memory address mapping 4-2
 - Non-volatile 4-1
 - Operating modes 4-8
 - Power-down mode 4-8
 - Program mode 4-8
 - Ready-to-read mode 4-8

Sector 4-3

- Flash program memory 3-1

G

- Gate disturb 4-6
- GPIO 11-1

H

- Hamming code 4-9
- High-impedance 11-2

I

- Idle mode 7-12, 7-17
- IEN0 13-13
- IIC 17-1–17-22
 - Baud rate generation 17-5
 - Bus arbitration 17-6
 - Clock synchronization 17-6
 - Master receive 17-10
 - Master transmit 17-7
 - Operating modes 17-7
 - Register description 17-15
 - Register map 17-15
 - Slave receive 17-13
 - Slave transmit 17-12
 - Software reset 17-7
 - Status code 17-4
- In-Application Programming 4-11, 4-12
 - Aborting background Flash erase 4-14
 - Background Flash erase 4-14
 - Background Flash program 4-13
 - Flash read mode status 4-16
 - Non-background Flash erase 4-14
 - Non-background Flash program 4-13
- In-System Programming 4-11
- Inter-IC Bus 17-1
- Internal data memory 3-2
- Internal RAM 3-1
- Interrupt 9-1
 - Flags 9-32
 - Handling 9-12
 - Priority 9-30
 - Registers 9-16

Keyword Index

Structure 9-10

Type 1 9-11

Type 2 9-11

Interrupt Response Time 9-13

L

LED and Touch-Sense Controller 19-1

LED Controller 19-1

LEDTSCU 19-1

FNCOL Interpretation 19-16

Function 19-5

Function Enable & Control 19-15

Interrupt 19-19

LED Driving 19-8

Pin Control 19-18

Pin Current Capability 19-10

Registers 19-20

Time-Multiplexed Function 19-14

Timing Calculations 19-16

Touch-Sensing 19-11

LIN 16-13-??

Baud rate detection 16-18

Baud rate range selection 16-16

Break field 16-14

Break/synch field detection 16-16

Header transmission 16-15

LIN frame 16-13

LIN protocol 16-13

Synch byte 16-14

M

MDU 12-1–12-14

Division 12-4

Division with single right shift 12-6

Error detection 12-6

Interrupt generation 12-7

Multiplication with single left shift 12-5

Normalize 12-5

Register description 12-8

Register map 12-8

Shift 12-5

Memory organization 3-1

Special Function Registers 3-4

Address extension by mapping 3-4

Mapped 3-4

Standard 3-4

Address extension by paging 3-7

Local address extension

3-7

Save and restore 3-8

Memory protection 3-4

Minimum erase width 4-3

Minimum program width 4-6

Multifold replications 4-4

Multiplication/Division Unit 12-1

O

OCDS 10-1

Breakpoint 10-8

External Break 10-10

Hardware 10-8

On Instruction Address

10-8

On IRAM Address 10-9

Software 10-10

Breakpoint Reaction 10-11

NMI 10-12

NMI Priority 10-13

OD 14-12

P

P0 register description 11-17

P1 register description 11-26

P2 register description 11-33

Parallel ports 11-1

General bidirectional port structure
11-3

Input mode 11-2

Kernel registers 11-7

Alternate input functions 11-10

Alternate output functions 11-10

Input control register 11-10

Open drain control register 11-8

Port data in register 11-8

Keyword Index

Port data out register 11-7
 Pull-Up/Pull-Down device register 11-9
 Px_ALTSELn **11-10**
 Px_OD **11-8**
 Px_PUDEN **11-9**
 Px_PUDSEL **11-9**
 Register addresses 11-5
 Normal mode 11-1
 Open drain mode 11-1
 P0 11-12
 P1 11-22
 P2 11-30
 PCON 7-24
 Peripheral clock management 7-21
 Personal computer host 4-11
 Power-down mode 7-12
 Power-down wake-up reset 7-7
 Power-on reset 7-2, 7-6
 Prewarning period 8-4
 Program memory 3-2

R

Read access time 4-1
 Register Overview 3-13
 Reset

Module behavior 7-8

RS-232 4-11

RTC 15-1

Basic Timer Operation 15-2

Mode 1 15-3

Mode 3 15-4

Oscillator 15-2

Registers Description 15-6

RTC registers 15-7

S

Schmitt trigger 11-2

Sectorization 4-2

Soft reset 7-7

Special Function Register area 3-1

SSC 18-1–18-27

Baudrate generation 18-15

Continuous transfer operation 18-14

Data width 18-9

Error detection 18-16

Baud rate error 18-17

Phase error 18-17

Receive error 18-17

Transmit error 18-18

Full duplex operation 18-10

Half duplex operation 18-13

Interrupts 18-16

Operating mode selection 18-8

Register description 18-20

Register map 18-20

Synchronous Serial Interface 18-1

T

TCON 13-10

THx 13-10

Timer 0 and Timer 1

Counter 13-1

External control 13-3

Mode 0, 13-bit timer 13-5

Mode 1, 16-bit timer 13-6

Mode 2, 8-bit automatic reload timer 13-7

Mode 3, two 8-bit timers 13-8

Register Description 13-9

Register map 13-9

Timer operations 13-3

Timer overflow 13-3

Timer 2 14-1

Auto-Reload mode 14-5

Up/Down Count Disabled 14-5

Up/Down Count Enabled 14-6

Capture mode 14-8

External interrupt function 14-10

Registers description 14-11

Timer operations 14-5

Timers 0 and 1

Registers

IEN0 **13-13**

TCON **13-10**

THx **13-10**

TLx **13-9**

TMOD **13-11**

TLx 13-9

TMOD 13-11

Touch-Sense Controller 19-1

U

UART 16-1–16-25

Baud rate 16-9

Baud rate generator 16-9

Interrupt requests 16-6

Mode 0, 8-bit shift register 16-3

Mode 1, 8-bit UART 16-4

Mode 2, 9-bit UART 16-6

Mode 3, 9-bit UART 16-6

Modes 16-3

Multiprocessor communication 16-8

Register description 16-19

Register map 16-19

User Identification 5-2

USER_ID 5-1, 6-15

W

Watchdog timer 8-1, 8-2

Module suspend control 8-3

Register description 8-7

Servicing 8-4

Watchdog timer reset 7-7

Window boundary 8-4

Wordline address 4-5

Write buffers 4-6

X

XRAM 3-1

Register Index

B

BCON 16-22
BGH 16-25
BGL 16-24

C

CCU6_CC60RH 20-46
CCU6_CC60RL 20-46
CCU6_CC60SRH 20-47
CCU6_CC60SRL 20-47
CCU6_CC61RH 20-46
CCU6_CC61RL 20-46
CCU6_CC61SRH 20-47
CCU6_CC61SRL 20-47
CCU6_CC62RH 20-46
CCU6_CC62RL 20-46
CCU6_CC62SRH 20-47
CCU6_CC62SRL 20-47
CCU6_CC63RH 20-83
CCU6_CC63RL 20-83
CCU6_CC63SRH 20-84
CCU6_CC63SRL 20-84
CCU6_CMPMODIFH 20-54
CCU6_CMPMODIFL 20-53
CCU6_CMPSTATH 20-52
CCU6_CMPSTATL 20-51
CCU6_IENH 20-121
CCU6_IENL 20-120
CCU6_INPH 20-125
CCU6_INPL 20-124
CCU6_ISH 20-113
CCU6_ISL 20-112
CCU6_ISRH 20-118
CCU6_ISRL 20-118
CCU6_ISSH 20-116
CCU6_ISSL 20-116
CCU6_MCMCTR 20-104
CCU6_MCMOUTH 20-109
CCU6_MCMOUTL 20-107
CCU6_MCMOUTSHH 20-106

CCU6_MCMOUTSL 20-106
CCU6_MODCTRH 20-99
CCU6_MODCTRL 20-98
CCU6_PISEL0H 20-128
CCU6_PISEL0L 20-127
CCU6_PISEL2 20-129
CCU6_PSLR 20-103
CCU6_T12DTCH 20-49
CCU6_T12DTCL 20-49
CCU6_T12H 20-43
CCU6_T12L 20-43
CCU6_T12MSELH 20-55
CCU6_T12MSELL 20-55
CCU6_T12PRH 20-44
CCU6_T12PRL 20-44
CCU6_T13H 20-79
CCU6_T13L 20-79
CCU6_T13PRH 20-81
CCU6_T13PRL 20-81
CCU6_TCTR0H 20-59
CCU6_TCTR0L 20-57
CCU6_TCTR2H 20-63
CCU6_TCTR2L 20-61
CCU6_TCTR4H 20-65
CCU6_TCTR4L 20-64
CCU6_TRPCTRH 20-101
CCU6_TRPCTRL 20-100
CHINCR 21-89
CHINFR 21-87
CHINSR 21-88
CPU

Registers

EO 2-5
CRCR1 21-23
CRMR1 21-21
CRPR1 21-24

E

EO 2-5
ETRCR 21-37
EVINCR 21-86

EVINFR 21-84
EVINPR 21-58
EVINSR 21-85
EXICON0 9-21, 9-22

F

FEAH 4-10
FEAL 4-10

G

GLOBCTR 21-14
GLOBSTR 21-15

H

HWBP0H 10-17
HWBP0L 10-17
HWBP1H 10-18
HWBP1L 10-17
HWBP2H 10-18
HWBP2L 10-18
HWBP3H 10-19
HWBP3L 10-19
HWBPDR 10-16
HWBPSR 10-16

I

IEN0 9-17
IEN1 9-18
IIC_ADDR 17-16
IIC_ADDRX 17-17
IIC_BRCCR 17-21
IIC_CNTR 17-18
IIC_DATA 17-17
IIC_SRST 17-21
IIC_STAT 17-20
INPCR0 21-51
IP 9-30
IP1 9-31
IPH 9-31
IPH1 9-32
IRCON0 9-24
IRCON1 9-24
IRCON2 9-25

IRCON3 9-26

L

LCBR 21-54
LINST 16-23
LTS_COMPARE 19-23
LTS_GLOBCTL0 19-21
LTS_GLOBCTL1 19-22
LTS_LDLINE 19-25
LTS_LDTSCTL 19-24
LTS_TSCTL 19-26
LTS_TSVL 19-27

M

MDU_MD4 12-11
MDU_MDUCON 12-12
MDU_MDUSTAT 12-14
MDU_MDx 12-10
MDU_MR4 12-11
MDU_MRx 12-10
MMICR 10-15
MMWR2 10-19
RTC_CNT0 15-8
RTC_CNT1 15-8
RTC_CNT2 15-9
RTC_CNT3 15-9
RTC_RTCCR0 15-10
RTC_RTCCR1 15-10
RTC_RTCCR2 15-11
RTC_RTCCR3 15-11
MODPISEL1 9-22, 16-2
MODPISEL2 13-2, 14-2

N

NMICON 9-19

P

P0_ALTSEL0 11-19
P0_ALTSEL1 11-20
P0_ALTSEL2 11-20
P0_DATAIN 11-17
P0_DATAOUT 11-17
P0_OD 11-18

P0_PUDEN 11-19
 P0_PUDSEL 11-18
 P1_ALTSEL0 11-28
 P1_ALTSEL1 11-28
 P1_DATAIN 11-26
 P1_DATAOUT 11-26
 P1_OD 11-27
 P1_PUDEN 11-28
 P1_PUDSEL 11-27
 P2_DATAIN 11-33
 P2_EN 11-33
 P2_PUDEN 11-34
 P2_PUDSEL 11-34
 PASSWD 3-11
 PCON 2-6, 16-22
 PMCON1 14-3, 19-4, 20-12, 21-3
 PORT_PAGE 11-5
 PRAR 21-41
 PSW 2-4
 Px_ALTSELn 11-10
 Px_DATAIN 11-8
 Px_DATAOUT 11-7
 Px_EN 11-11
 Px_OD 11-8, 11-9
 Px_PUDEN 11-9

Q

Q0R0 21-34
 QBUR0 21-36
 QINR0 21-33
 QMR0 21-29
 QSR0 21-31

R

RCRx (x = 0 - 3) 21-77
 RESRxH (x = 0 - 3) 21-72
 RESRxL (x = 0 - 3) 21-69, 21-72, 21-75
 RTC_RTCON 15-7

S

SBUF 16-20
 SCON 16-20
 SCU_PAGE 7-25

SDCON 7-4
 SSC_BRH 18-26
 SSC_BRL 18-26
 SSC_CONH 18-22, 18-24
 SSC_CONL 18-21, 18-24
 SSC_RBL 18-27
 SSC_TBL 18-27
 SYSCON0 3-6

T

T2_RC2H 14-15
 T2_RC2L 14-15
 T2_T2CON 14-13
 T2_T2CON1 14-14
 T2_T2H 14-16
 T2_T2L 14-16
 T2_T2MOD 14-12
 TCON 9-23, 9-27

V

VFCR 21-79

W

WDTCON 8-8, 23-27, 23-28, 23-29
 WDTL 8-9
 WDTL 8-9
 WDTREL 8-7
 WDTWINB 8-9

X

XADDRH 3-3

www.infineon.com